Taylor & Francis
Taylor & Francis Group

Mathematical and Computer Modelling of Dynamical Systems

Methods, Tools and Applications in Engineering and Related Sciences

ISSN: 1387-3954 (Print) 1744-5051 (Online) Journal homepage: https://www.tandfonline.com/loi/nmcm20

Automated generation of hybrid automata for multi-rigid-body mechanical systems and its application to the falsification of safety properties

E.M. Navarro-López & M.D. O'Toole

To cite this article: E.M. Navarro-López & M.D. O'Toole (2018) Automated generation of hybrid automata for multi-rigid-body mechanical systems and its application to the falsification of safety properties, Mathematical and Computer Modelling of Dynamical Systems, 24:1, 44-75, DOI: 10.1080/13873954.2017.1369437

To link to this article: https://doi.org/10.1080/13873954.2017.1369437

MATHEMATICAL AND COMPUTER MODELLING OF DYNAMICAL SYSTEMS, 2018
VOL. 24, NO. 1, 44–75
https://doi.org/10.1080/13873954.2017.1369437

Taylor & Francis
Taylor & Francis Group

Automated generation of hybrid automata for multi-rigid-body mechanical systems and its application to the falsification of safety properties

E.M. Navarro-López [a] and M.D. O'Toole[b]

[a]School of Computer Science, The University of Manchester, Manchester, UK; [b]School of Electrical and Electronic Engineering, The University of Manchester, Manchester, UK

ABSTRACT

What if we designed a tool to automatically generate a dynamical transition system for the formal specification of mechanical systems subject to multiple impacts, contacts and discontinuous friction? Such a tool would represent an advance in the description and simulation of these complex systems. This is precisely what this paper offers: Dyverse Rigid Body Toolbox (DyverseRBT). This tool requires a sufficiently expressive computational model that can accurately describe the behaviour of the system as it evolves over time. For this purpose, we propose an alternative abstraction of multi-rigid-body (MRB) mechanical systems with multiple contacts as an extended version of the classical hybrid automaton, which we call MRB hybrid automaton. One of the chief characteristics of the MRB hybrid automaton is the inclusion of computation nodes to encode algorithms to calculate the contact forces. The computation nodes consist of a set of non-dynamical discrete locations, discrete transitions and guards between these locations, and resets on transitions. They can account for the energy transfer not explicitly considered within the rigid-body formalism. The proposed modelling framework is well suited for the automated verification of dynamical properties of realistic mechanical systems. We show this by the falsification of safety properties over the transition system generated by DyverseRBT.

ARTICLE HISTORY
Received 21 June 2016
Accepted 16 August 2017

KEYWORDS
Hybrid systems; hybrid automata models; design automation; computational methods; computer simulation

1. Introduction

1.1. *Motivation and scope of the problem*

This paper proposes a novel computational modelling framework that facilitates the automated generation of transition systems of complex mechanical systems for which a fully and clear description is difficult to obtain. We model mechanical systems with multiple impacts, contacts and discontinuous friction as hybrid dynamical systems, specifically, as hybrid automata [1–4], which provide a suitable modelling framework for the specification and analysis of complex hybrid systems – where continuous and discrete, smooth and abrupt parts interact with each other.

The automated generation of the dynamic relationships and discrete transitions of the systems treated here are possible by the proposal of a computational model – called the multi-rigid-body (MRB) hybrid automaton – that can accurately describe the behaviour of the system as it evolves over time. A method of converting a certain class of MRB problems into an MRB hybrid

CONTACT E.M. Navarro-López eva.navarro@manchester.ac.uk School of Computer Science, The University of Manchester, Oxford Road, Kilburn Building, Manchester M13 9PL, UK
Supplemental data for this article can be accessed here.

automaton is proposed. The derived hybrid automaton expresses the rigid-body system in a standard form with a number of special elements termed computation nodes. The contribution of this paper is to provide an alternative solution to the problem of the formal specification of complex dynamic interactions in MRB systems, with the ultimate goal of the automated generation of a discrete description to be able to simulate the system's dynamic behaviour. By building on previous mechanical models for MRB systems with multiple impacts, contacts and discontinuous friction [5–14], we offer a higher level of abstraction to specify the multiple transitions (dynamic and discrete) on these systems. These transitions and dynamic relationships grow with an increasing number of rigid bodies and multiple contacts and can get unmanageable and not clearly specified with classical mechanical models. This is overcome with the MRB hybrid automaton and its automated generation.

Although there are restrictions and limitations in the systems treated in this paper (see Section 1.2), they are broad enough to validate this paper as a novel and worthwhile contribution for the hybrid system modelling and engineering communities. For example, the growing interest in formal methods and hybrid systems in the robotics community is manifested in a wide range of recent works [15]. Formal specification is also well suited for autonomous vehicle control [16], robot motion planning and control [17,18] and control of semiautonomous vehicles [19]. Furthermore, the modelling and analysis framework built in this paper is applicable to a broader class of networked complex systems, as it has been recently shown for the case of neuronal networks [20].

Creating computational models of MRB systems is not straightforward. The governing dynamic equations are piecewise smooth and contain discontinuities and jumps in the state space resulting from changing friction and impacts. Moreover, the governing equations themselves may be non-linear and contain transcendental functions. The possible occurrence of multiple contacts further expounds the problem.

Hybrid automaton is a powerful computational-oriented framework for specifying the dynamic behaviour of rigid-body mechanical systems. Thus far, however, work has mainly focused on examples with only single points of contact. Consider for instance, the ubiquitous bouncing-ball example (see Ref. [21] and references therein). Modelling mechanical systems using hybrid automata becomes more difficult when there is the possibility of multiple contacts. Consider two separate objects sharing a set of contact points along their respective surfaces. Now, consider a third object which collides with one of the two original objects. We know that the collision, despite being separate, will affect the behaviour at the site of the other contacts between the first two objects. Physically, we know this to be caused by energy transfer occurring over extremely short timescales. However, when we model mechanical systems, we often simplify the problem by assuming rigid bodies that neglect the microscopic and fast-acting internal physical processes. The transition of energy from one contact site to the next is assumed instantaneous, and the mechanism by which this process occurs is invisible to us. We must instead resort to non-analytical numerical procedures, such as linear and non-linear complementarity or proximal point methods, to obtain solutions to the contact forces [5–7,9,11,13,14,22].

In this paper, we propose to model mechanical systems as hybrid automata by supporting multiple contacts in the system that occurs simultaneously. The hybrid automaton assumes that Newton's impact law (restitution) and Coulomb friction are acting at the points of contact. Multiple contacts are addressed by introducing what we call computation nodes. This is one of the key new elements that we add to the classical hybrid automaton modelling framework. Computation nodes are a collection of non-dynamical discrete states, transitions, guards and reset functions which are used to find valid combinations of contact forces for each configuration of the mechanical system. The combination of all these elements can account for the energy transfer not explicitly considered within the rigid-body formalism. In the computation node, the executions of the hybrid automaton are assumed to operate over zero time as the discrete states are non-dynamical. The physical analogue of this is the fast-acting dynamics acting at a

microscopic level which we assume to occur instantaneously in the rigid-body mechanics formalism. The inclusion of the computation nodes gives us the flexibility to encode more complex and even iterative algorithms into the hybrid-automaton framework, with the benefit that we can model more complex mechanical systems with multiple contacts.

The resulting computational–mechanical model is referred to as the MRB hybrid automaton, which consists of two types of discrete locations: dynamical discrete locations – the classical discrete locations in a hybrid automaton, that is locations with an associated dynamical system – and non-dynamical discrete locations – the new type of discrete locations introduced in our modelling framework. This automaton provides a complete and precise modelling framework and fully describes the transitions, transition conditions and operating modes present in the class of mechanical systems under study. The preliminary idea of the MRB hybrid automaton was proposed in other work [23]. The MRB hybrid automaton is a modification of the discontinuous dynamical systems (DDSs) hybrid automaton [24] – which models DDSs with discontinuous state derivatives (e.g. systems with dry friction) – to include jumps in the states (a discontinuity usually associated with systems with impacts). These modifications are inspired by mythical modes in discontinuous systems, originally proposed in Ref. [25] and further expanded to hybrid systems by Refs. [26,27].

The hybrid automaton modelling framework proposed to model mechanical systems with multiple impacts and friction has two main potential applications. First, the generation of event-driven simulations of MRB mechanical systems. Second, the MRB hybrid automaton is well suited for the formal verification of dynamical properties of realistic mechanical systems. Due to the complexity involved, verifying these systems manually is laborious and could be eased by automatic computational techniques.

The automatic generation of MRB hybrid automata is a necessary step in the formal analysis of a complex mechanical system. Moreover, we highlight its useful application in the simulation of MRB systems, in particular with respect to event-driven schemes. For this purpose, we have produced a tool called Dyverse Rigid Body Toolbox (DyverseRBT). The acronym Dyverse corresponds to DYnamically driven VERification of Systems with Energy considerations, and RBT corresponds to Rigid Body Toolbox. DYVERSE is a novel computational–dynamical framework for the modelling, analysis and control of complex systems [28]. DyverseRBT automatically generates the MRB hybrid automaton and its simulation from a mechanical specification given by the user. In other words, DyverseRBT produces a dynamical transition system that defines the computational semantics of the MRB system.

In the system description given, the events or discrete transitions are defined in order to clearly specify all the possible changes in the dynamics. As a consequence, the hybrid automaton proposed may be easily translate to a program or to any other description language, as it is proposed, for instance, in Refs. [29–33]. We note that the hybrid automaton formulation can be interpreted as an event-driven scheme (or event-tracking time-stepping scheme) for the numerical time-integration of the system; however, an improvement of event-driven schemes is out of the scope of this paper. The alternative numerical time-integration techniques to event-driven schemes are the time-stepping schemes (also known as event-capturing time-stepping schemes). These schemes are not based on the accurate detection of events, are quite robust in contact detection and also have convergence proofs. The reader is referred to Refs. [5,14,34,35] for further details and references on time-stepping schemes for mechanical systems.

In the last two decades, there has been an effort in proposing different semantics and computational-oriented frameworks for modelling systems exhibiting sliding-type behaviour [36,37] and discrete events. For example, object-oriented models [38,39], hybrid dynamic models [24,27,40–48] and approximate bisimulations [49–52] are successfully used for different applications.

The approach proposed here follows a similar rationale to the recent work by Mosterman et al. [53], in the sense that we create computational semantics of the dynamical behaviour of a

system which is described by differential equations and works in several modes of operation. The ultimate goal is using computer science methods (like model checking) for analysis purposes, in addition to computing simulations of continuous-time-based models. We highlight that Ref. [53] proposes a general design framework for stiff hybrid systems with special emphasis on control synthesis. Our work could be extended to be used for control synthesis, but it is out of the scope of this paper.

The results presented in this paper must be distinguished from the different simulators for hybrid and cyber-physical systems like Stateflow of Simulink® [54], Modelica [55], Ptolemy [47] or CyPhySim [56]. However, since the MRB hybrid automaton fits within the standard automaton framework, it should be possible to implement the derived hybrid automaton on other hybrid system simulators. A by-product of the automated MRB hybrid automaton generation has been the automated creation of Simulink *S*-functions. These *S*-functions implement the hybrid automaton and can be dropped into the Simulink models to interact with other components. Given this, a similar approach could be followed for other simulator tools.

To explore the potential application of our MRB modelling framework and the tool DyverseRBT in automated verification, we produce a method to check in an automated way that some properties are not held for a family of MRB mechanical systems which are subject to multiple contacts, impacts and discontinuous friction and interpreted as MRB hybrid automata. Particularly, we follow an *engineering-driven* approach, with the main goal of finding bugs in the design rather than proving the correctness of the system. Indeed, from a practical viewpoint, engineering and control designers are interested in automated verification techniques that can provide counterexample trajectories – that is, trajectories violating a given property – in order to consider scenarios difficult to extract from simulations. This is typically referred to as falsification (finding errors), which is different to proving that errors do not exist. In this paper, the analysis of properties in MRB mechanical systems will be done through falsification. Dynamical restrictions will be specified as safety properties ('something bad will never happen'), which will be ultimately reduced to check the satisfiability of Boolean formulas.

The idea of falsification is not new in dynamical systems, see for example Refs. [57–60] where continuous-time non-linear smooth systems are studied. Particularly, in Ref. [59], there is a focus on the validation of controllers. We highlight the works of Refs. [61–65] on falsification of safety properties in hybrid systems, and the Simulink®-based tool S-TaLiRo for hybrid systems with relatively simple dynamics in each mode or subsystem [66].

The implementation of our especially tailored falsification method for MRB systems is done through the pilot software Dyverse Bounded Model Checker (DyverseBMC), especially designed for mechanical systems with multiple impacts and friction. The input of DyverseBMC is a dynamical transition system generated by DyverseRBT. The approach we adopt for falsification in this paper will mainly use bounded model checking (BMC) [67,68]. In BMC, instead of computing the states that satisfy a specification (property), a SAT (Boolean satisfiability) solver is used to find an error, typically known as a counterexample, which is a trace that does not satisfy the specification. Hence, a property can be assessed not to be satisfied if no counterexample is found; specifically, if the SAT procedure cannot generate a propositional formula that is satisfiable for a counterexample. In DyverseBMC, instead of following the traditional approach of using SAT-based BMC [69,70], we use a *lazy-SMT* (Satisfiability Modulo Theories) solving approach inspired by the recent works of Refs. [71–74]. SMT-based BMC is becoming increasingly important in practical applications and hybrid systems, for example iSAT [75,76], MathSAT [77], Absolver [78,79], verification of networks of linear hybrid automata [80], bounded reachability analysis for linear hybrid automata [81] and the recent bounded model-checker dReach [82] that uses the SMT solver dReal [83,84]. Whilst some of these tools (iSAT, Absolver and dReach with dReal) can be mainly applied to non-linear dynamical systems and some types of hybrid systems, they cannot be directly applied to our systems – non-linear hybrid automata, with a high number of different types of discrete locations – in their present form.

Even though the falsification technique proposed and implemented here via DyverseBMC is not optimized and has limitations, still it represents a valuable solution, considering that not many current automated verification and falsification techniques would be able to be applied without important expansions to the type of systems our MRB hybrid automaton is modelling.

In conclusion, this paper proposes a new way of modelling and simulating of dynamical behaviours of mechanical systems with multiple impacts, contacts and discontinuous friction within the framework of hybrid automata, which is applicable to the analysis of dynamical properties in an automated way. We formulate for the first time a class of MRB mechanical systems with friction and impacts as an expanded type of hybrid automata. The consideration of computation nodes within hybrid automata opens the possibility to explicitly include in the model the computation of key parameters and dynamic relationships for complex systems, with a clear application on the automated generation of computational models for these systems. The conceptual idea of this alternative abstraction of hybrid systems has potential to influence not just the modelling of mechanical systems, but also the modelling and analysis of complex systems with complex computations that cannot be clearly encoded in classical hybrid automata. As an evidence of this statement, we have recently applied the idea of the MRB hybrid automaton and the computation nodes for the modelling of complex neuronal networks [20]. The computation nodes are used to calculate and specify the synaptic weights and the synaptic currents for single neurons or populations of neurons. We think that it is important to separate these computations from the membrane dynamics of neurons. The inclusion of the computation nodes gives us the flexibility to encode more complex and even iterative algorithms into the hybrid-automaton framework, which has the benefit that we can model non-trivial neuronal interactions. A summary of the contribution of this paper can be found in Figure 1.

1.2. *Assumptions and restrictions*

The approach outlined in this paper can, in principle, be applied to the analysis of a broad range of rigid-body systems. However, in the present work, we impose a number of assumptions to simplify the systems under test. We do so as a practical measure, to make the problem solvable and to reduce the corresponding complexity of our analysis tools. We outline these assumptions and restrictions here as follows:

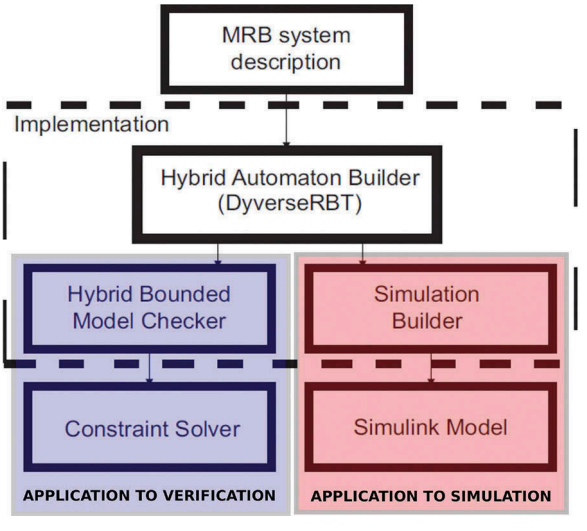

Figure 1. Summary of the contribution of this paper and the applications of the proposed modelling framework.

- *Mechanical-rigid-body systems with a small number of objects*. The method requires the construction of a complete map of all possible contact combinations to completely define the MRB hybrid automaton. This can create very large automata due to combinatorial explosion, which can be impractical to handle. This will be better understood in the examples of Section 3.8.2 and in Table 2, where several multiple contact problems are presented. Thus, rigid-body systems with only a small number of objects (and contact combinations) can be practically studied.
- *Mechanical-rigid-body systems consisting of spheres*. Our definition of the MRB hybrid automaton only accounts for rigid bodies with spherical surfaces. It would be straightforward to extend the definition to other shapes and surfaces; however, we do not do so in the present work. Restricting to spheres keeps the possible number of contact combinations low – that is, each pair of objects can share only one possible contact – and thus reduces the size of the automata to a more manageable size.
- *Our hybrid automaton formulation is an event-driven scheme*, and as such, is ill-equipped to handle finite accumulations of non-smooth events (like impacts or stick/slip transitions).

1.3. *Motivating example: automatic generation of the MRB hybrid automaton by DyverseRBT*

An example is introduced to facilitate the description of our MRB hybrid automaton construction. We will return to this example at each stage, building up the complexity as we progress.

The example here is the *Interception Game* which is shown in Figure 2. The game involves two balls. The first ball (ball 1, the upper ball in the figure) is unconstrained and is subject only to gravity. The second ball (ball 2, the lower ball) is driven by a proportional feedback controller which constrains the ball to the horizontal plane ($y = 0$) and tracks the position of ball 1 along the x and z axis. The objective of the game is to use ball 2 to *intercept ball 1* as it falls towards $y = 0$ and bounce it back into free space. If ball 2 does not intercept, then ball 1 falls through the horizontal plane. If ball 1 falls through the plane within some bounded time, then the game is lost.

Define $\mathbf{x}_1 = \left(x_1, y_1, z_1, \alpha_1, \beta_1, \gamma_1\right)^T$ as the state vector for ball 1, where x_1, y_1 and z_1 are positions along their respective axes, and α_1, β_1 and γ_1 are rotations about the x, y and z axes, respectively. Similarly, define $\mathbf{x}_2 = \left(x_2, y_2, z_2, \alpha_2, \beta_2, \gamma_2\right)^T$ for ball 2.

The system starts in a non-contact state and is governed by the following dynamic equations:

$$\begin{aligned} \ddot{y}_1(t) &= -g, \\ \ddot{y}_1(t) &= -g - 100y_2(t), \\ \ddot{x}_2(t) &= -K(x_2(t) - x_1(t)), \\ \ddot{z}_2(t) &= -100(z_2(t) - z_1(t)), \end{aligned} \tag{1}$$

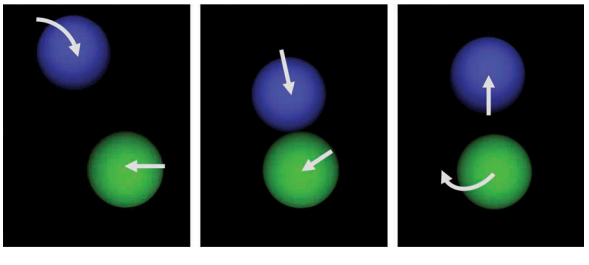

Figure 2. Interception game.

where the double dot denotes second derivative with respect to time t, g is the force generated by gravity and K is the proportional feedback gain. The accelerations for the remaining states is set to zero. We also define a set of initial conditions:

$$I = \left\{ \begin{array}{c} \mathbf{x}_1(0) = (0,3,0,0,0,0)^T, \\ \mathbf{x}_2(0) = (1.5,0,0,0,0,0)^T, \\ \dot{\mathbf{x}}_1(0) = (1,1,0,0,0,0)^T, \\ \dot{\mathbf{x}}_2(0) = (0,0,0,0,0,0)^T \end{array} \right\},$$

where the dot denotes derivative with respect to time.

The MRB hybrid automaton is automatically generated by DyverseRBT. The package DyverseRBT uses a Matlab-based program to generate an MRB hybrid automaton of an MRB mechanical system from a minimal set of information on a rigid-body system: a list of entities and their interacting forces. The package can then use the automaton to generate an *S*-function of the mechanical system for Simulink simulations. This is a potentially powerful domain-specific application, and to our knowledge, nothing similar exists to date.

A schematic of the package is shown in Figure 3. The user inputs a text file consisting of information on a set of rigid bodies, the forces interacting between entities, and external inputs as shown in Figure 4 for the example of the interception game. This file has the extension .drbt. The user also creates a Simulink model containing an empty *S*-function block. The external inputs in the text file become the inputs to the *S*-function block in the Simulink model. A function called 'DyverseCreate' reads the input file, extracts the relevant information and generates a hybrid automaton in the form of a Matlab structure, following the formalism that will be defined in the next sections. The hybrid automaton is then converted into C code for an *S*-function which is compiled using the on-board MEX compiler. Thus, the package generates an *S*-function for the Simulink model (sim.dll), and a text file to display the automata (display.txt). The *S*-function can be dropped into a Simulink model and used as an ordinary block. More details are given when we come back to this example in Section 3.8.

The programs that form DyverseRBT and the input files for the examples used in this paper are available at http://staff.cs.manchester.ac.uk/navarroe/research/dyverse/DyverseRBT_BMC.zip.

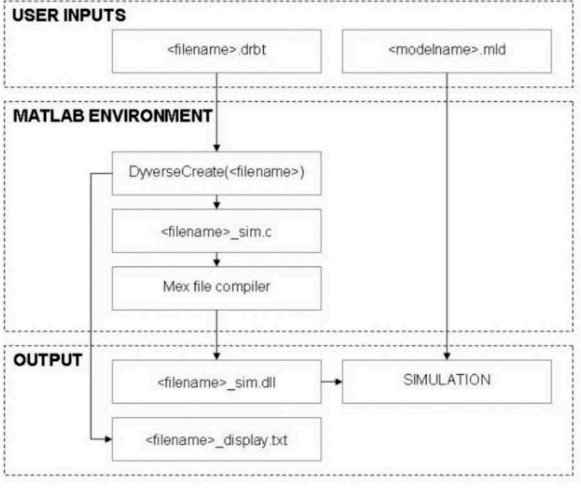

Figure 3. Outline of DyverseRBT.

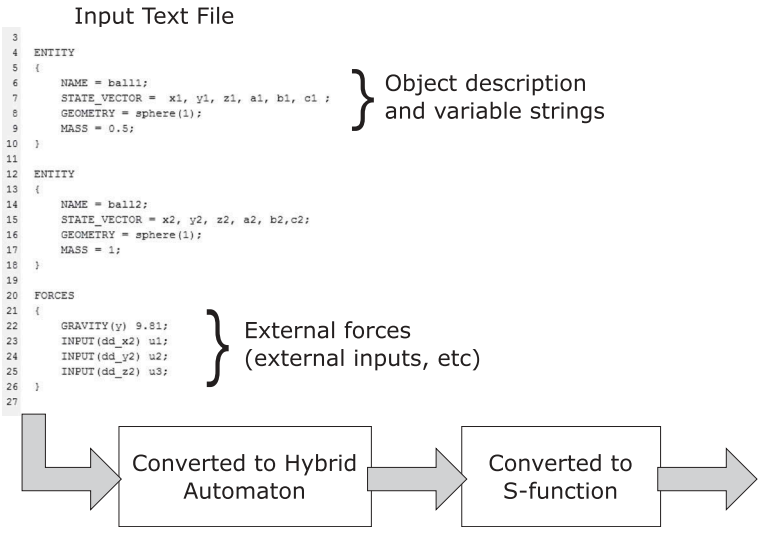

Figure 4. Input file to DyverseRBT provided by the user for the interception game.

2. Mechanics

The primary element in our modelling framework based on the MRB hybrid automaton is the rigid-body object or 'entity', which will include the position, velocity and acceleration in the contact reference frame, in addition to geometrical and physical properties of each body. The notation used is primarily drawn from [13] and will, as far as possible, be kept the same. In the rest of the paper, the subscripts N and T will denote normal and tangential. T as a superscript on a vector or matrix will denote transpose.

Definition 2.1: (Entity) An entity $\mathcal{E}_i$ is a representation of a rigid body consisting of the collection,

$$\mathcal{E}_i = (q_i, m_i, J_i(q_i)),$$

where

- $q_i = \left(\bar{q}_i^T, \alpha_i, \beta_i, \gamma_i\right)^T$ is a vector containing the position $\bar{q}_i \in \mathbb{R}^3$ of the centre of rotation of the entity in a global Cartesian coordinate frame, and the rotation of the entity about the x-axis (α_i), about the y-axis (β_i) and about the z-axis (γ_i) of the global frame.
- m_i is the total mass of the entity.
- $J_i : \mathbb{R}^6 \to \mathbb{R}$ is a strictly convex differentiable function, called the surface function, which defines the surface of the entity as $J_i(q_i) = 0$.

The condition of strict convexity of J_i means that any pair of entities can only share one contact point. This simplifies the synthesis of the MRB hybrid automaton in the next section. Note that this restriction does not preclude the multiple contact case. An entity can have more than one contact point provided that each of the contact points is with a different entity – that is, each pair of entities shares only one contact point. The condition of differentiability is necessary to ensure that we can calculate a unique solution for the surface normal at any point along the surface. This is critical for our treatment of contact problems.

The conditions on the function J_i are heavily restrictive on the types of rigid-body problems we can consider. For instance, the conditions imply that we are only able to address rigid bodies or entities with shapes such as spheres or ellipsoids, but not those with corners or concave shapes. However, it is possible, in principle, to extend what follows to relax these conditions, for example

by allowing an entity to consist of multiple surface functions, having rounded corners etc. We do not do so in the present work in order to maintain simplicity.

From now and so on, $|\cdot|$ denotes the absolute value for any real number, or the cardinality of a set, and $|\cdot|$ is the 2-norm on the Euclidean space $\mathbb{R}^n$. Furthermore, to ease the notation, in the ordinary differential equations (ODEs) presented, we will not write the dependency on time of the variables.

Consider the case of two entities $\mathcal{E}_1, \mathcal{E}_2$ about to come into contact as shown in Figure 5. Denote this point of contact the ith point of contact: that is, we assume that there are already $i-1$ points of contact present in our system of rigid bodies. Denote p_1 as a point in the global reference frame that lies on the surface of $\mathcal{E}_1$, and p_2 similarly as a point on the surface of $\mathcal{E}_2$. The points are chosen so that when collision occurs, $p_1 = p_2$. We can define a *gap function* $g_{N,1}$ which returns the distance between the two points in a direction normal to the surface of $\mathcal{E}_1$,

$$g_{N,1} = \mathbf{n}_1 \cdot (p_2 - p_1), \tag{2}$$

where $\mathbf{n}_1$ is a vector normal to the surface of $\mathcal{E}_1$ at the point p_1. We can put the relative velocity and acceleration between the surface points in terms of entity coordinates q_1, q_2 as,

$$\dot{g}_{N,1} = \mathbf{w}_{N,1}^T \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix} + \dot{\mathbf{n}}_1^T (p_2 - p_1), \tag{3}$$

$$\ddot{g}_{N,1} = \mathbf{w}_{N,1}^T \begin{pmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{pmatrix} + w_{a^N,1}(q_1, q_2, \dot{q}_1, \dot{q}_2), \tag{4}$$

where

$$\mathbf{w}_{N,1}^T = \begin{bmatrix} -\mathbf{n}_1^T & -\mathbf{n}_1^T \tilde{r}_1 & \mathbf{n}_1^T & \mathbf{n}_1^T \tilde{r}_2 \end{bmatrix},$$

and

$$w_{a,N,1} = \begin{bmatrix} -2\dot{\mathbf{n}}_1^T & -2\dot{\mathbf{n}}_1^T \tilde{r}_1 & 2\dot{\mathbf{n}}_1^T & 2\dot{\mathbf{n}}_1^T \tilde{r}_2 \end{bmatrix},$$

and $\tilde{r}_i$, for all i, is a skew-symmetric matrix. We shall use N in the subscript from here on to indicate that the term refers to the normal component. Note that $\dot{\mathbf{n}}_1^T (p_2 - p_1) = 0$ at the point of contact ($p_1 = p_2$).

This can be generalized for an MRB system consisting of M entities $\{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_M\}$ with $I = \{1, 2, \ldots, i, \ldots, P\}$ points of contact between them occurring at any one time. Let $q = \left[q_1^T, q_2^T, \ldots, q_M^T\right]^T$ be a vector containing the individual coordinates of each entity. For the ith contact, we can define

$$\dot{g}_{N,i} = \mathbf{w}_{N,i}^T \dot{q}, \tag{5}$$

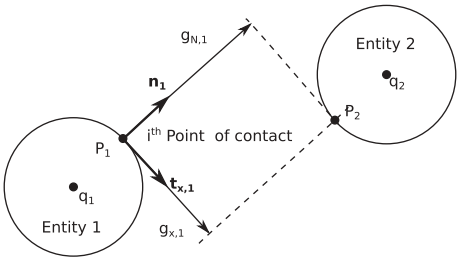

Figure 5. Two entities about to contact.

$$\ddot{g}_{N,i} = \mathbf{w}_{N,i}^T \ddot{q} + w_{a,N,i}(q, \dot{q}). \tag{6}$$

The following terms are introduced:

$$\dot{g}_N = \left(\dot{g}_{N,1}, \dot{g}_{N,2}, \ldots, \dot{g}_{N,P}\right)^T,$$

$$\mathbf{W}_N = \left(\mathbf{w}_{N,1}, \mathbf{w}_{N,2}, \ldots, \mathbf{w}_{N,P}\right),$$

$$\mathbf{w}_{a,N}(q, \dot{q}) = \left(w_{a,N,1}, w_{a,N,2}, \ldots, w_{a,N,P}\right)^T.$$

Then, for the whole system consisting of M entities and P contacts:

$$\dot{g}_N = \mathbf{W}_N^T \dot{q}, \tag{7}$$

$$\ddot{g}_N = \mathbf{W}_N^T \ddot{q} + \mathbf{w}_{a,N}. \tag{8}$$

The resulting Equations (2), (7) and (8) are useful for expressing the constraint that no two rigid-bodies can occupy the same space, i.e. penetration cannot occur.

We now consider the constraints acting tangentially to the contact surface. Returning to the single contact in Equation (2), define two vectors $\mathbf{t}_{x,1}, \mathbf{t}_{y,1}$ orthogonal to one another and to the normal vector $\mathbf{n}_1$. The combination of the three vectors forms a coordinate reference frame at the site of contact. We refer to this frame as the contact frame. The position of point p_2 relative to p_1 projected onto the tangential plane is

$$g_{T,1} = \begin{pmatrix} g_{x,1} \\ g_{y,1} \end{pmatrix} = \begin{pmatrix} \mathbf{t}_{x,1} \\ \mathbf{t}_{y,1} \end{pmatrix} \cdot (p_2 - p_1). \tag{9}$$

Following the same procedure as for the normal component, we can generalize for the ith contact and obtain expressions for the relative tangential velocity and acceleration,

$$\dot{g}_{T,i} = \mathbf{w}_{T,i}^T \dot{q}, \tag{10}$$

$$\ddot{g}_{T,i} = \mathbf{w}_{T,i}^T \ddot{q} + w_{a,T,i}. \tag{11}$$

For the whole system, we obtain

$$\dot{g}_T = \mathbf{W}_T^T \dot{q}, \tag{12}$$

$$\ddot{g}_T = \mathbf{W}_T^T \ddot{q} + \mathbf{w}_{a,T}, \tag{13}$$

where

$$\dot{g}_T = \left(\dot{g}_{T,1}, \dot{g}_{T,2}, \ldots, \dot{g}_{T,P}\right)^T,$$

$$\mathbf{W}_T = \left(\mathbf{w}_{T,1}, \mathbf{w}_{T,2}, \ldots, \mathbf{w}_{T,P}\right),$$

$$\mathbf{w}_{a,T} = \left(w_{a,T,1}, w_{a,T,2}, \ldots, w_{a,T,P}\right)^T.$$

The subscript T from here-on refers to the tangential component.

There are two forces acting at the point of contact. The first is the reaction force, which prevents any two rigid bodies from occupying the same space, that is, penetration between entities cannot occur. The second force is friction, which resists motion along the surface of the point of contact.

The reaction force $\lambda_{N,i}$ acts along the surface normal to the point of contact and is always sufficiently large to enforce the unilateral constraints $g_{N,i} = 0$, $\dot{g}_{N,i} = 0$ and $\ddot{g}_{N,i} = 0$ at the point of contact. We refer to these as the non-penetration constraints or Signorini's contact law. The reaction force can be expressed by the complementarity condition on the acceleration level as,

$$\begin{cases} \lambda_{N,i} \geq 0 & \text{if } \ddot{g}_{N,i} = 0, \\ \lambda_{N,i} = 0 & \text{if } \ddot{g}_{N,i} > 0, \end{cases}$$

which yields to the so-called contact linear complementarity problem (LCP) [8]. We also refer the reader to Chapter 4 of Ref. [85] for an explanation of complementarity systems under the framework of hybrid dynamical systems.

Friction acts on the plane tangential to the point of contact $\lambda_{T,i} = \left(\lambda_{x,i}, \lambda_{y,i}\right)^T$ where $\lambda_{x,i}$ is in the direction of $g_{x,i}$ and $\lambda_{y,i}$ is in the direction of $g_{y,i}$. Here, we use the Coulomb friction given by the friction law in Ref. [86].

The friction force is bounded such that $\left\|\lambda_{T,i}\right\| < \mu_i \lambda_{N,i}$ where μ_i is the coefficient of friction and can either be a constant or a function of the relative velocity. The contact can be in one of three modes: *stick, trans* or *slip* mode. In *stick mode*, the friction force $\lambda_{T,i}$ is any value within the bound such that the relative acceleration along the surface of the contact point is constrained to zero, that is, the bilateral constraint $\left\|\ddot{g}_{T,i}\right\| = 0$ is enforced. In *slip mode*, the friction force is on the bound, acting in a direction opposing the velocity. The transitional-slip mode is the same as the slip mode but where the friction force acts in a direction opposing acceleration rather than velocity. Hence, the friction force is determined by,

$$\begin{aligned} &\left\{\lambda_{T,i} : \left\|\lambda_{T,i}\right\| \leq \mu_i \lambda_{N,i}\right\} \quad \text{if } \begin{matrix} \left\|\ddot{g}_{T,i}\right\| = 0, \\ \left\|\dot{g}_{T,i}\right\| = 0, \end{matrix} \quad (\textit{stick}) \\ &\lambda_{T,i} = -\operatorname{sgn}\left(\ddot{g}_{T,i}\right)\mu_i \lambda_{N,i} \quad \text{if } \begin{matrix} \left\|\ddot{g}_{T,i}\right\| > 0, \\ \left\|\dot{g}_{T,i}\right\| = 0, \end{matrix} \quad (\textit{trans}) \\ &\lambda_{T,i} = -\operatorname{sgn}\left(\dot{g}_{T,i}\right)\mu_i \lambda_{N,i} \quad \text{if } \left\|\dot{g}_{T,i}\right\| > 0. \ (\textit{slip}) \end{aligned}$$

Impact energy transitions can be modelled using Newton's restitution law [11]:

$$\dot{g}^+_{N,i} = -e_{N,i}\dot{g}^-_{N,i},\ 0 \leq e_{N,i} \leq 1,$$

$$\dot{g}^+_{T,i} = -e_{T,i}\dot{g}^-_{T,i},\ \left\|e_{T,i}\right\| \leq 1,$$

where $\dot{g}^+_{N,i}, \dot{g}^+_{T,i}$ are the post-impact velocities and $\dot{g}^-_{N,i}, \dot{g}^-_{T,i}$ are the pre-impact velocities. The terms $e_{N,i}, e_{T,i}$ are the normal and tangential restitution coefficients, respectively.

We can create complementarity conditions similar to the contact force laws. Define $\Lambda_{N,i}$ as an impulse along the surface normal to the ith contact. Then, based on the non-penetration constraint, we have

$$\begin{cases} \Lambda_{N,i} \geq 0 & \text{if } \dot{g}^+_{N,i} + e_{N,i}\dot{g}^-_{N,i} = 0, \\ \Lambda_{N,i} \geq 0 & \text{if } \dot{g}^+_{N,i} + e_{N,i}\dot{g}^-_{N,i} > 0, \end{cases}$$

and in the tangential plane,

$$\begin{cases} \left\{\Lambda_{T,i} : \left\|\Lambda_{T,i}\right\| \leq \mu_i \Lambda_{N,i}\right\} & \text{if } \dot{g}^+_{T,i} + e_{T,i}\dot{g}^-_{T,i} = 0, \\ \Lambda_{T,i} = -\operatorname{sgn}(\dot{g}_{T,i})\mu_i \Lambda_{N,i} & \text{if } \left\|\dot{g}^+_{T,i} + e_{T,i}\dot{g}^-_{T,i}\right\| > 0. \end{cases}$$

This impact law has, in general, poor prediction qualities when comparing with experimental observation. However, the law is simple and is very numerically and computationally tractable [12].

In this paper, we will use the lower case λ interchangeably to refer to both contact forces and impulses (Λ). It will be obvious to the reader which we are referring to, force or impulse, from the context of its use and the supporting discussion. By choosing λ to represent both force and impulse, we remove redundant notation and avoid the unnecessary duplication of some of the equations we are about to introduce.

3. The MRB hybrid automaton

In this section, we will describe the components of our modelling framework, mainly the elements of the MRB hybrid automaton, the contact list and the contact combination graph. These latter two components are essential for the definition of the possible configurations and contacts that may occur in the system. The elements of our hybrid-automaton-based framework and their interrelationships that will be described in the following subsections can be interpreted as an event-driven scheme for the numerical time-integration of a type of mechanical systems with multiple impacts, contacts and discontinuous friction. Here, we invite the reader to check well-known event-driven schemes that have been specifically designed for the numerical time-integration of non-smooth mechanical systems, especially Chapter 8 of Ref. [5].

The MRB hybrid automaton is based on the hybrid model given in Ref. [3,4] and the DDSs hybrid automaton proposed in Ref. [24]. For the sake of simplifying the notation, we will not consider inputs and outputs for the basic MRB hybrid automaton.

3.1. *The general MRB hybrid automaton*

The MRB hybrid automaton has the following general definition. The details of its elements are explained in the subsequent sections.

Consider a system consisting of M entities with P *possible contacts*. The MRB hybrid automaton is a collection,

$$H_{\mathrm{MRB}} = (S, E, \mathcal{X}, \mathrm{Dom}, \mathcal{F}, \mathrm{Init}, G, R),$$

with:

- S a finite set of **discrete locations**. We will have two types of discrete locations: **dynamical discrete locations** (the classical discrete locations in a hybrid automaton, that is, locations with an associated dynamical system) and **non-dynamical discrete locations** – the new type of discrete locations introduced in our modelling framework – which are grouped into **computation nodes**. The reader is referred to Sections 3.2, 3.3 and 3.4 for more details. The set of dynamical discrete locations is denoted by S_{d}, and the set of non-dynamical discrete locations is denoted by S_{non}, and $S = S_{\mathrm{d}} \bigcup_{S_{\mathrm{non}}}$.
- $\mathcal{X} \subseteq \mathbb{R}^{12\mathrm{M}+3\mathrm{P}}$ is the **continuous state space**. The continuous state vector is a generalized coordinate vector $q \in \mathbb{R}^{6\mathrm{M}}$ and a generalized velocity vector $\dot{q} \in \mathbb{R}^{6\mathrm{M}}$ for all the M entities in the system, plus $\lambda_N = \{\lambda_{N,1}, \lambda_{N,2}, \ldots, \lambda_{N,P}\} \in \mathbb{R}^P$ and $\lambda_T = \left\{\lambda_{T,1}^T, \lambda_{T,2}^T, \ldots, \lambda_{T,P}^T\right\} \in \mathbb{R}^{2P}$ as the continuous vectors of contact forces, acting at each of the P possible contacts. That is, for $x \in \mathcal{X}$, $x = (q, \dot{q}, \lambda_N, \lambda_T)^T$.
- $E \subseteq S \times S$ represents the set of **discrete transitions** or **edges** in H_{MRB}, which is finite. The discrete transitions are deterministic and are of different types, being between dynamical

discrete locations, non-dynamical discrete locations and combinations of both; they are specified in Section 3.5.

- $\mathrm{Dom} : S_\mathrm{d} \to 2^{\mathcal{X}}$ are the **dynamical discrete location domains**. Dom assigns a set of continuous states to each dynamical discrete location of S_d, thus, for $s \in S_\mathrm{d}$, $\mathrm{Dom}(s) \subseteq \mathcal{X}$. These domains are formally specified in Section 3.6.
- $\mathcal{F}$ represents the **continuous dynamics**: $\mathcal{F} = \{f_s(x) : s \in S_\mathrm{d}\}$ is a collection of vector fields such that $f_s : \mathcal{X} \to \mathcal{X}$ describes the dynamics in each dynamical discrete location. Each $f_s(x)$ with $x \in \mathcal{X}$ is Lipschitz continuous on $\mathcal{X}$ in order to ensure that in each dynamical discrete location, s the solution exists and is unique.
- $\mathrm{Init} \subseteq S_\mathrm{d} \times \mathcal{X}$ is a **set of initial states**.
- G represents the **guard maps** for each discrete transition. $G : E \to 2^{\mathcal{X}}$. G assigns to each edge $e \in E$ a set of continuous states ($G(e) \subset \mathcal{X}$) which enables transitions along that edge. The guards are of different types depending on the type of discrete transition they are associated with, being between dynamical discrete locations, non-dynamical discrete locations and combinations of both. More details are given in Section 3.5.
- R denotes the **reset maps**, which define the re-initialization of the continuous state every time a discrete transition takes place. All the resets associated with discrete transitions between dynamical discrete locations do not change the continuous state. The resets between non-dynamical discrete locations within computation nodes depend on the numerical procedure which uses the dynamical discrete location models. More details are given in Section 3.7.

The formal definitions of a hybrid time trajectory and an execution of the MRB hybrid automaton on such a trajectory are similar to the ones given for classical hybrid automata and can be found in Ref. [3]. In general terms, the consideration of computation nodes does not change the definition of the hybrid time trajectory and the execution of the hybrid automaton, since time does not evolve in the computation nodes. However, these nodes are considered as nodes that perform some necessary computations, and roughly speaking, 'they guide the executions' of the hybrid automaton to the relevant dynamical discrete locations based on the information of the contact list and the contact combination graph (described in the next section), and consequently, they might be considered as part of the dynamical discrete locations of the MRB hybrid automaton. The influence of the computation nodes on the evolution of the dynamics is formally specified in the sections below.

3.2. *Contact list and contact combination graph*

In this section, we will define the contact list and the contact combination graph, which are necessary concepts to later define the dynamical and non-dynamical discrete locations in Sections 3.3 and 3.4.

The set of all gap functions $g_{N,i}$ for every *possible contact* that could occur in the system is denoted by Γ_N. Let $\dot{\Gamma}_N = \left\{\dot{g}_{N,1}, \dot{g}_{N,2}, \ldots\right\}$ and $\ddot{\Gamma}_N = \left\{\ddot{g}_{N,1}, \ddot{g}_{N,2}, \ldots\right\}$ be the sets of successive derivatives of the gap functions with respect to time.

We assign to each gap function in the set Γ_N an index or label and define $\bar{I}$ as the collection of all these labels. Thus, $i \in \bar{I} \Leftrightarrow g_{N,i} \in \Gamma_N$. Note that the set $\bar{I}$ naturally labels the elements of Γ_N, $\dot{\Gamma}_N$ and $\ddot{\Gamma}_N$. That is, for some $i \in \bar{I}$, we have $g_{N,i} \in \Gamma_N$, $\dot{g}_{N,i} \in \dot{\Gamma}_N$ and $\ddot{g}_{N,i} \in \ddot{\Gamma}_N$.

As an example, take the three-ball problem in Figure 6. There are three possible contacts and thus three gap functions: $g_{N,ab}$ between balls a and b, $g_{N,bc}$ between balls b and c and $g_{N,ac}$ between balls a and c. Thus, $\bar{I} = \{ab, bc, ac\}$ and $\Gamma_N = \left\{g_{N,ab}, g_{N,bc}, g_{N,ac}\right\}$.

The *contact combination graph* is a structure that contains information on the possible configurations the system of entities could take, and whether it is possible for the system to evolve from one configuration to another. Each vertex of the contact combination graph

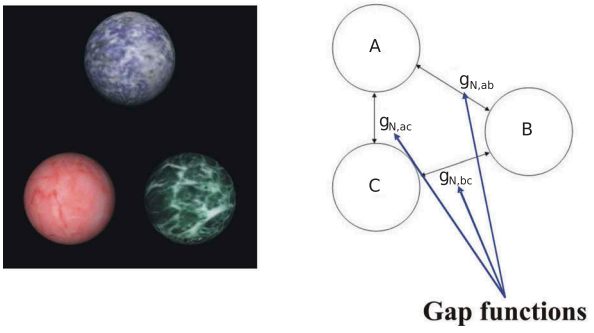

Figure 6. Three-ball problem: associated gap functions.

represents some unique combination of closed and open contacts which could possibly occur at some time in our system. A closed contact is when two surfaces are touching – they are actually in contact – and an open contact is when they are not. Each edge represents a possible transition from one set of closed and open contacts to another. Then, the contact combination graph is denoted by $\Omega = (V, E_\Omega)$, where V is a set of vertices and E_Ω a set of edges.

For each vertex $V_k \in V$, we define $I_k \subseteq 2^{\bar{I}}$ as the set of indices of all closed contacts when the contact combination graph is in the kth vertex,

$$I_k = \left\{ j \in \bar{I} : \, g_{N,j} \in \Gamma_N(q), \, g_{N,j} \leq 0 \right\},$$

with q as defined in Section 2. Each set of indices will form a *configuration* of the system. If it is possible for the system to directly evolve from configuration I_k, to a configuration I_l, then there is an edge between V_k and V_l, that is $(k, l) \in E_\Omega$. From here on, we will refer to any specific contact combination by its index. That is, when the contact combination graph is in the kth vertex, we say that the system is in the kth contact combination. Further, as a convention, we will reserve $k = 0$ for the contact combination where no contacts occur ($I_k = \{\emptyset\}$). Finally, we denote $N_k = |I_k|$ as the number of contacts occurring in the kth contact combination.

Returning to the three-ball example, let us say that V_1 is associated with the configuration $I_1 = \{ab, ac\}$, shown on the left of Figure 7, V_2 is associated with $I_2 = \{ab, ac, bc\}$, shown on the upper right, and V_3 with $I_3 = \{ab, bc\}$, shown on the lower right. There is an edge between V_1 and V_2 because it is possible for the three balls to change from one configuration to the other. For the same reason, there is an edge between V_2 and V_3. However, the three balls cannot change directly from V_1 to V_2 without being in some intermediate configuration; therefore, there is no edge between these two.

3.3. *Dynamical discrete locations and associated dynamics*

For each vertex $V_k \in V$, there is a set of 3^{N_k} dynamical discrete locations: $S_k = \left\{ s_{k,1}, s_{k,2}, \ldots, s_{k,3^{N_k}} \right\}$. Each dynamical discrete location represents a different arrangement of stick, slip or transition mode at each of the contact points. For example, the vertex V_1 in our three-ball example has two contacts. These contacts could both be in stick mode, or both in slip mode, or one in trans mode, the other in stick mode, and so on. Each combination (stick-stick, slip-stick etc.) is represented by a different dynamical discrete location and the ensuing dynamics are governed by a different continuous dynamical equation.

To each location $s_{k,i}$, we associate the following index sets:

- $I_{\mathrm{st},k,i}$ the indices of all contacts where friction is in *stick* mode:

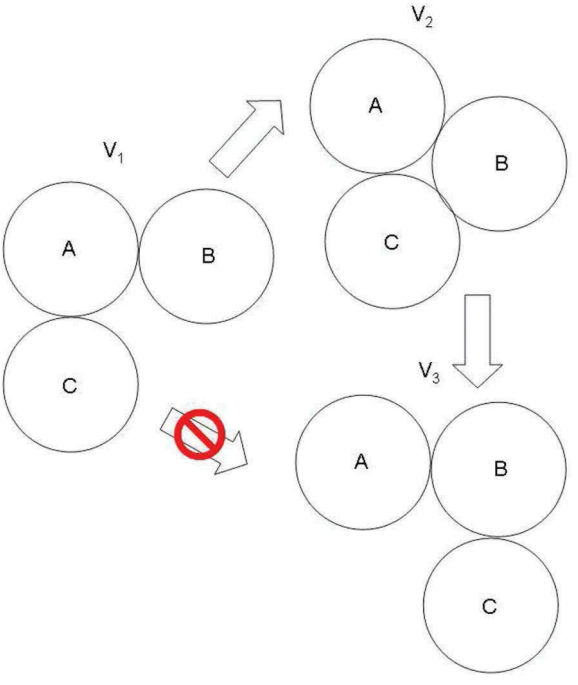

Figure 7. Three-ball problem: building the contact combination graph.

$$I_{\mathrm{st},k,i} = \left\{ j \in I_k : \left\| \dot{g}_{T,j} \right\| = 0 \wedge \left\| \ddot{g}_{T,j} \right\| = 0 \right\}.$$

- $I_{\mathrm{tr},k,i}$ the indices of all contacts where friction is in *trans* mode:

$$I_{\mathrm{tr},k,i} = \left\{ j \in I_k : \left\| \dot{g}_{T,j} \right\| = 0 \wedge \left\| \ddot{g}_{T,j} \right\| > 0 \right\}.$$

- $I_{\mathrm{sl},k,i}$ the indices of all contacts where friction is in *slip* mode:

$$I_{\mathrm{sl},k,i} = \left\{ j \in I_k : \left\| \dot{g}_{T,j} \right\| > 0 \right\}.$$

The set $S_{\mathrm{d}} = \bigcup_{V_k \in V} S_k$. We also note that $I_{\mathrm{st},k,i} \cup^{I_{\mathrm{tr},k,i}} \cup^{I_{\mathrm{sl},k,i}} = I_k$ and $I_{\mathrm{st},k,i} \cap^{I_{\mathrm{tr},k,i}} \cap^{I_{\mathrm{sl},k,i}} = \{\emptyset\}$.
For each dynamical discrete location $s_{k,i}$, we have a continuous dynamical system of the form:

$$\mathbf{M}(q)\ddot{q} = \mathbf{h}(q, \dot{q}) + \left(\mathbf{W}_{N,k}(q) + \mathbf{W}_{\mathrm{tr},k,i}(q, \ddot{q}) + \mathbf{W}_{\mathrm{sl},k,i}(q, \dot{q}) \right) \lambda_N + \mathbf{W}_{\mathrm{st},k,i}(q) \lambda_T, \tag{14}$$

where $\mathbf{M}(q)$ is the mass matrix, $\mathbf{h}(q, \dot{q})$ is a force vector containing all conservative forces, input forces, and Coriolis terms, and

$$\mathbf{W}_{N,k} = \left(\ldots, \mathbf{W}_{N,j}, \ldots \right), \quad \lambda_N = \left(\ldots, \lambda_{N,j}, \ldots \right)^T, \qquad \forall j \in I_k,$$

$$\mathbf{W}_{\mathrm{st},k,i} = \left(\ldots, \mathbf{W}_{T,j}, \ldots \right), \quad \lambda_T = \left(\ldots, \lambda_{T,j}^T, \ldots \right)^T, \qquad \forall j \in I_{\mathrm{st},k,i},$$

$$\mathbf{W}_{\mathrm{tr},k,i} = \left(\ldots, -\mathbf{W}_{T,j} \frac{\ddot{g}_{T,j}^T}{\left\| \ddot{g}_{T,j} \right\|} \mu_j \left(\dot{g}_{T,j} \right), \ldots \right), \quad \forall j \in I_{\mathrm{tr},k,i},$$

$$\mathbf{W}_{\mathrm{sl},k,i} = \left(\ldots, -\mathbf{W}_{T,j} \frac{\dot{g}_{T,j}^T}{\left\| \dot{g}_{T,j} \right\|} \mu_j \left(\dot{g}_{T,j} \right), \ldots \right), \quad \forall j \in I_{\mathrm{sl},k,i}.$$

The contact forces λ_N, λ_T are computed using the following equation,

$$\begin{pmatrix} \lambda_N \\ \lambda_T \end{pmatrix} = \begin{pmatrix} \mathbf{W}_{N,k}^T \mathbf{M}^{-1} \mathbf{W}_{N,k} & \mathbf{W}_{N,k}^T \mathbf{M}^{-1} \mathbf{W}_{\mathrm{st},k,i} \\ \mathbf{W}_{\mathrm{st},k,i}^T \mathbf{M}^{-1} \mathbf{W}_{N,k} & \mathbf{W}_{\mathrm{st},k,i}^T \mathbf{M}^{-1} \mathbf{W}_{\mathrm{st},k,i} \end{pmatrix}^+ \begin{bmatrix} \mathbf{W}_{N,k}^T \mathbf{M}^{-1} \mathbf{h} + w_{a,N} \\ \mathbf{W}_{\mathrm{st},k,i}^T \mathbf{M}^{-1} \mathbf{h} + w_{a,T} \end{bmatrix}. \tag{15}$$

The $^+$ indicates the pseudo-inverse. If there are no redundant constraints, then the matrix on the right-hand side becomes non-singular and the pseudo-inverse can be replaced by a conventional matrix inversion.

3.4. *Non-dynamical discrete locations and computation nodes*

The MRB hybrid automaton includes a new type of discrete locations, referred to as non-dynamical discrete locations. The collection of these discrete locations, and the edges, guards and resets between them, is termed a computation node, whose main elements are given in Table 1. The discrete locations in the computation nodes do not have any continuous-time dynamics.

The computation nodes are used to model the numerical methods that affect the dynamics of the system, but which cannot themselves be represented as dynamical elements. To clarify, in our MRB hybrid automaton, we use the computation nodes to model the numerical procedures used in calculating contact forces at impact, or the state of the contact forces (stick, slip etc.) when in sustained contact. In the framework of rigid-body dynamics, we often assume that changes in contact forces are instantaneous. Otherwise, we must model the fast-acting microscopic dynamics, which is computationally wasteful and largely unnecessary in a macroscopic analysis. Thus instead, we use iterative or LCP-based numerical procedures to calculate the contact forces. LCP stands for linear complementarity problem.

The computation nodes are inspired by the well-known *mythical modes*, originally proposed in Ref. [25] and later expanded to hybrid systems by Ref. [26,27]. They have to be understood in this sense, just a set of transition intermediate states associated to a discontinuous change. They do not have any continuous dynamics. Further, one might consider the physical analogue of the computation nodes as being the fast-acting dynamics operating at the microscopic scale which we conventionally assume to act instantaneously in the framework of rigid-body mechanics.

For each vertex $V_k \in \{V_k \in V : N_k > 0\}$, there is a set of non-dynamical locations:

$$\mathcal{I}_k = \{\mathcal{I}_{k,\mathrm{en}}, \mathcal{I}_{k,1}, \ldots, \mathcal{I}_{k,\mathrm{ex}}\},$$

$$\mathcal{C}_k = \{\mathcal{C}_{k,\mathrm{en}}, \mathcal{C}_{k,1}, \ldots, \mathcal{C}_{k,\mathrm{ex}}\},$$

Table 1. Main elements of the computation nodes.

Purpose	Elements	Types
To compute contact forces	Set of non-dynamical discrete locations Entrance discrete location (all edges going into the computation node end here) Exit discrete location (all edges to locations outside the node leave from here) Edges and guards between non-dynamical discrete locations Reset functions to find contact forces on transitions	Impact computation nodes Contact computation nodes

which are called the *impact computation node* and the *contact computation node*, respectively. Both computation nodes have similar form. The sets $\mathcal{I}_k$ and $\mathcal{C}_k$ consist of an entrance discrete location with subscript en, an exit discrete location with subscript ex and a set of discrete locations in between which are used for the computation of the contact forces. The number of these discrete locations varies in number depending on the algorithm used to calculate the contact forces. The edges going into the computation node are received by the entrance discrete location, and all edges leaving the computation node leave from the exit discrete location. The edges connecting the intermediate locations are explained in Section 4. We highlight that every different computation node will have a different number of non-dynamical discrete locations.

The algorithm we use to compute the contact forces is the successive over-relaxation proximal point method (SORPROX) as described in Ref. [14]. The SORPROX method operates by computing some new estimate of the contact force at some point of contact based on some previous initial guesses. If it is a feasible contact force, that is, it satisfies the complementarity conditions in Section 2, then the new estimate is accepted and becomes the next iteration of the contact force. Otherwise, the next iteration is chosen to be the feasible point nearest the new estimate. A new estimate is then produced for the next contact force, and so on, repeatedly cycling through and iterating the contact forces until they converge to a solution.

An outline of the overall computation node is shown in Figure 8. For simplicity, let us say we are in the ith impact node with the contact index set $I_i = \{1, 2, 3, \ldots, n\}$. Recall from our earlier definition that the contact index is an index of all closed contacts in a given configuration. In this case, there are n contact points, indexed $1, 2, 3, \ldots, n$.

When an impact occurs, the entrance location $\mathcal{I}_{i,\text{en}}$ becomes active. The execution of the computation node traverses each of the intermediate locations, computing new contact forces, until the nth node $\mathcal{I}_{i,1} \to \mathcal{I}_{i,2} \to \mathcal{I}_{i,3} \to \cdots \to \mathcal{I}_{i,n}$. The location $\mathcal{I}_{i,n}$ has an edge going to the exit node and an edge going to the entrance node. If the all the constraints are satisfied, then a transition $\mathcal{I}_{i,n} \to \mathcal{I}_{k,\text{ex}}$ occurs, and the state vector q is reset. If the constraints are not satisfied, the transition $\mathcal{I}_{i,n} \to \mathcal{I}_{i,\text{en}}$ takes place and the execution of the computation node is repeated to determine a new set of contact forces.

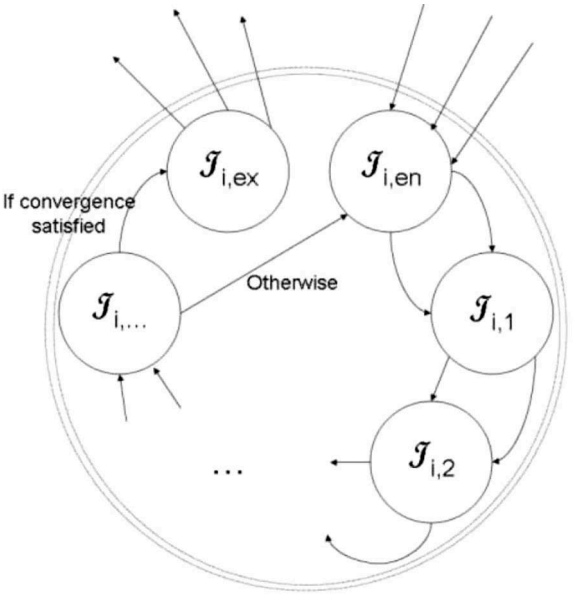

Figure 8. Approximate outline of an example impact computation node.

In this paper, we use computation nodes based on the SORPROX method [14] for calculating contact forces. We have previously reported details of the computation node for SORPROX in Ref. [23]. For the sake of brevity, we will omit repeating these details in this paper and refer the interested reader to the referenced work.

3.5. *Edges and guards*

The set of edges of H_{MRB} consists of the union of the following:

- For all k such that $\{V_k \in V \,:\, N_k > 0\}$:
$$\left\{\left(\mathcal{I}_{k,\mathrm{ex}}, \mathcal{C}_{k,\mathrm{en}}\right)\right\}.$$
- For all k and j such that $(k,j) \in E_\Omega$ and $\left(N_k > N_j \wedge N_j > 0\right)$:
$$\left\{\left(\mathcal{I}_{k,\mathrm{ex}}, \mathcal{C}_{j,\mathrm{en}}\right), \ldots\right\}.$$
- For all k and i such that $\left\{V_k \in V : N_k > 0 \wedge \left|I_{\mathrm{tr},k,i}\right| = 0\right\}$:
$$\left\{\left(s_{k,i}, \mathcal{C}_{k,\mathrm{en}}\right), \ldots\right\}.$$
- For all k and i such that $\left\{V_k \in V : N_k > 0 \wedge \left(\left|I_{\mathrm{st},k,i}\right| > 0 \vee \left|I_{\mathrm{tr},k,i}\right| > 0\right)\right\}$:
$$\left\{\left(\mathcal{C}_{k,\mathrm{ex}}, s_{k,i}\right), \ldots\right\}.$$
- For all k, j and i such that $\left\{V_k \in V : N_k > 0 \wedge \left|I_{\mathrm{tr},k,i}\right| > 0 \wedge \left|I_{\mathrm{tr},k,j}\right| = 0 \wedge \left(I_{\mathrm{tr},k,i} \cup^{I_{\mathrm{sl},k,i}} = I_{\mathrm{sl},k,j}\right)\right\}$:
$$\left\{\left(s_{k,i}, s_{k,j}\right), \ldots\right\}.$$
- For all k, j and i such that $(k,j) \in E_\Omega$, $\left(N_k < N_j\right)$ and $s_{k,i} \in S_k$:
$$\left\{\left(s_{k,i}, \mathcal{I}_{j,\mathrm{en}}\right), \ldots\right\}.$$
- For all k, j and i such that $(k,j) \in E_\Omega$, $\left(N_k > N_j \wedge N_j > 0\right)$ and $s_{k,i} \in S_k$:
$$\left\{\left(s_{k,i}, \mathcal{C}_{j,\mathrm{en}}\right), \ldots\right\}.$$
- If a discrete location S_0 such that $N_0 = 0$ exists, then for all $\{(k,0) \in E_\Omega\}$:
$$\left\{\left(\mathcal{I}_{k,\mathrm{ex}}, S_0\right), \left(S_0, \mathcal{I}_{k,\mathrm{en}}\right)\right\}$$
and for all i such that $s_{k,i} \in S_k$:
$$\left\{\left(s_{k,i}, S_0\right), \ldots\right\}$$

The guards for each of the edges in H_{MRB} are as follows:

$$G\left(\mathcal{I}_{k,\mathrm{ex}}, \mathcal{C}_{j,\mathrm{en}}\right) = \left\{x \in \mathcal{X} : \left(\forall n \in I_j, \dot{g}_{N,n} = 0\right) \wedge \left(\forall n \in I_k \backslash I_j, \dot{g}_{N,n} > 0\right)\right\},$$

$$G\left(\mathcal{I}_{k,\mathrm{ex}}, \mathcal{C}_{k,\mathrm{en}}\right) = \left\{x \in \mathcal{X} : \forall n \in I_k, \dot{g}_{N,n} = 0\right\}.$$

Let us define $\Pi_k = \underset{m:(k,m)\in E_\Omega}{\cup} I_m$. For all $i \in \{1, 2, \ldots, N_k\}$:

$$
\begin{aligned}
G(s_{k,i}, \mathcal{C}_{k,\text{en}}) &= \{x \in \mathcal{X} : (\forall j \in I_k, \lambda_{N,j} \geq 0) \\
&\wedge \left(\forall j \in \Pi_k \backslash I_k, g_{N,j} > 0 \vee \dot{g}_{N,j} > 0\right) \\
&\wedge \left(\left(\exists j \in I_{\text{st},k,i}, \mu_j \lambda_{N,j} < \left\|\lambda_{T,j}\right\|\right) \vee \left(\exists j \in I_{\text{sl},k,i}, \left\|\dot{g}_{T,j}\right\| = 0\right)\right)\}, \\
G(\mathcal{C}_{k,\text{ex}}, s_{k,i}) &= \left\{x \in \mathcal{X} : \left(\forall j \in I_{\text{st},k,i}, \left\|\ddot{g}_{T,j}\right\| = 0\right)\right. \\
&\wedge \left(\forall j \in I_{\text{tr},k,i}, \left\|\ddot{g}_{T,j}\right\| > 0\right) \\
&\left.\wedge \left(\forall j \in I_{\text{sl},k,i}, \left\|\dot{g}_{T,j}\right\| > 0\right)\right\}, \\
G(s_{k,i}, s_{k,j}) &= \left\{x \in \mathcal{X} : \left(\forall n \in I_{\text{tr},k,i}, \left\|\dot{g}_{T,n}\right\| > 0\right)\right. \\
&\wedge \left(\forall n \in \Pi_k \backslash I_k, g_{N,n} > 0 \vee \dot{g}_{N,n} > 0\right) \\
&\left.\wedge (\forall n \in I_k, \lambda_{N,n} \geq 0)\right\}, \\
G(s_{k,i}, \mathcal{I}_{j,\text{en}}) &= \left\{x \in \mathcal{X} : \left(\forall n \in I_j, g_{N,n} \leq 0 \wedge \dot{g}_{N,n} \leq 0\right)\right. \\
&\left.\wedge \left(\forall n \in \Pi_k \backslash I_j, g_{N,n} > 0 \vee \dot{g}_{N,n} > 0\right)\right\}, \\
G(s_{k,i}, \mathcal{C}_{j,\text{en}}) &= \{x \in \mathcal{X} : (\forall n \in I_j, \lambda_{N,n} \geq 0) \\
&\wedge (\forall n \in I_k \backslash I_j, \lambda_{N,n} < 0)\},
\end{aligned}
$$

where $\lambda_{N,j}, \lambda_{T,j}$ are jth rows of the vectors λ_N, λ_T given in equation (15). Finally, the following guards are specific only to S_0,

$$
\begin{aligned}
G(s_{k,i}, S_0) &= \{x \in \mathcal{X} : \forall j \in I_k, \lambda_{N,j} < 0\}, \\
G(\mathcal{I}_{k,\text{ex}}, S_0) &= \left\{x \in \mathcal{X} : \forall i \in I_k, \dot{g}_{N,i} > 0\right\}.
\end{aligned}
$$

These formula for the edges and guards can be better understood by applying them to specific examples. We refer the reader to Section 3.8 where some examples are presented, and to check the programs that form DyverseRBT and the input files for the examples used in this paper, which are available at http://staff.cs.manchester.ac.uk/navarroe/research/dyverse/DyverseRBT_BMC.zip.

3.6. *Location domains of dynamical discrete locations*

The location domains Dom are only considered for dynamical discrete locations, $s_{k,i} \in S_\text{d}$:

$$
\begin{aligned}
\text{Dom}(s_{k,i}) &= \{q \in \mathbb{R}^{6M} : (\forall j \in I_k, \lambda_{N,j} \geq 0) \\
&\wedge \left(\forall j \in \Pi_k \backslash I_k, g_{N,j} > 0 \vee \dot{g}_{N,j} > 0\right) \\
&\wedge \left(\forall j \in I_{\text{st},k,i}, \left\|\dot{g}_{T,j}\right\| = 0 \wedge \left\|\lambda_{T,j}\right\| \leq \mu_j \lambda_{N,j}\right) \\
&\wedge \left(\forall j \in I_{\text{tr},k,i} \left\|\dot{g}_{T,j}\right\| = 0 \wedge \left\|\ddot{g}_{T,j}\right\| > 0\right) \\
&\wedge \left(\forall j \in I_{\text{sl},k,i} \left\|\dot{g}_{T,j}\right\| > 0\right)\}.
\end{aligned}
$$

3.7. *Resets*

We highlight that all the resets associated with transitions between dynamical discrete locations do not change the continuous state. The resets between non-dynamical locations within

computation nodes depend on the numerical procedure which uses the dynamical discrete location models, and we refer to this type of resets as non-general resets, because they cannot be described with a general expression. For brevity, we do not include a description of the computation node we use in the present work. However, it is straightforward to derive it – including their resets – from the SORPROX method. We refer the reader to Ref. [23] for more details.

For the sake of clarity and to ease the specification of the model, we are taking some liberties with notation and do not employ the typical nomenclature of resets in hybrid automata.

Although we suggest that the resets in the computation nodes are non-general, we make two exceptions within the MRB hybrid automaton for which we give an expression for the resets. For an impact computation node consisting of the collection of the non-dynamical locations $\{\mathcal{I}_{k,\text{en}}, \mathcal{I}_{k,1}, \ldots, \mathcal{I}_{k,N}, \mathcal{I}_{k,\text{ex}}\}$, there is a reset on the edge $\mathcal{I}_{k,N} \to \mathcal{I}_{k,\text{ex}}$,

$$R(\mathcal{I}_{k,N}, \mathcal{I}_{k,\text{ex}}, x) \Rightarrow \dot{q} \to \mathbf{M}^{-1}(\mathbf{W}_{N,k}\lambda_N + \mathbf{W}_{T,k}\lambda_T) + \dot{q}.$$

Similarly, for a contact computation node consisting of locations $\{\mathcal{C}_{k,\text{en}}, \mathcal{C}_{k,1}, \ldots, \mathcal{C}_{k,N}, \mathcal{C}_{k,\text{ex}}\}$, there is a reset on the edge $\mathcal{C}_{k,N} \to \mathcal{C}_{k,\text{ex}}$,

$$R(\mathcal{C}_{k,N}, \mathcal{C}_{k,\text{ex}}, x) \Rightarrow \ddot{q} \to \mathbf{M}^{-1}(\mathbf{h} + \mathbf{W}_{N,k}\lambda_N + \mathbf{W}_{T,k}\lambda_T).$$

3.8. *The interception game example and multiple contact problems*

3.8.1. *MRB hybrid automaton for the interception game*

Now, we return to our motivating example introduced in Section 1.3. The MRB hybrid automaton for the interception game is shown in Figure 9. The guards, resets, and vector fields are omitted for clarity. The automaton features four dynamical locations: S_0 for the free-state (i.e. the balls are not in contact with one another), $s_{1,1}$ when the balls are in contact and in slip mode (i.e. they are

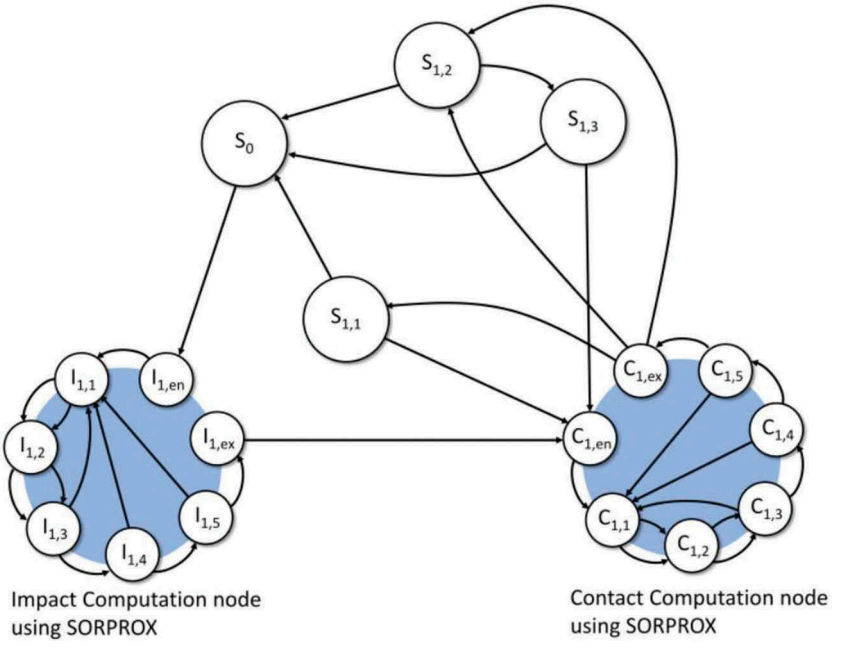

Figure 9. Hybrid automaton of the interception game.

slipping across each other), $s_{1,2}$ for contact and transition mode (or trans mode, see Section 3.3), and $s_{1,3}$ for contact and stick mode.

The impact and contact computation nodes are shown with their non-dynamical locations. The computation nodes are shown in the form for resolving contact forces using the SORPROX method. The first transition $\mathcal{I}_{1,\text{en}} \to \mathcal{I}_{1,1}$ (or equivalently $\mathcal{C}_{1,\text{en}} \to \mathcal{C}_{1,1}$) sets an initial guess for the two contact forces – then normal contact force, and the tangential friction force. The remaining transitions make new approximations of the contact forces and then test to see whether they have converged. If they have not converged, the cycle is repeated ($\mathcal{I}_{1,5} \to \mathcal{I}_{1,1}$ or $\mathcal{C}_{1,5} \to \mathcal{C}_{1,1}$); otherwise, the appropriate resets are applied on the state vector and the computation node is exited.

An MRB hybrid automaton is constructed for the interception game example using the DyverseRBT tool. The input file given by the user contains two entities (ball 1 and ball 2), each with a state vector, a chosen mass and a surface function for a sphere. This file was displayed in Figure 4. The input forces $u1$, $u2$ and $u3$ are external forces which can be input manually.

The implementation generates the contact list, the contact combination graph, locations, and computation nodes of an MRB hybrid automaton for the interception game model. It also generates the vector fields and domains of each location, and the edges with their respective guards and resets.

The hybrid automaton is then converted into *C* code for an *S*-function which is compiled using the on-board MEX compiler. The *S*-function can be dropped into a Simulink model and used as an ordinary block. The external inputs ($u1$, $u2$ and $u3$) given in the text file of Figure 4 become the inputs to the *S*-function block in the Simulink model. All this process is automated and the result is shown in Figure 10.

3.8.2. *Multiple contact problems*

We also consider the automatic construction of the MRB hybrid automaton from rudimentary rigid-body models consisting of 1, 2, 3 and 4 balls using the DyverseRBT tool. The case of 3 balls was considered in Section 3.2.

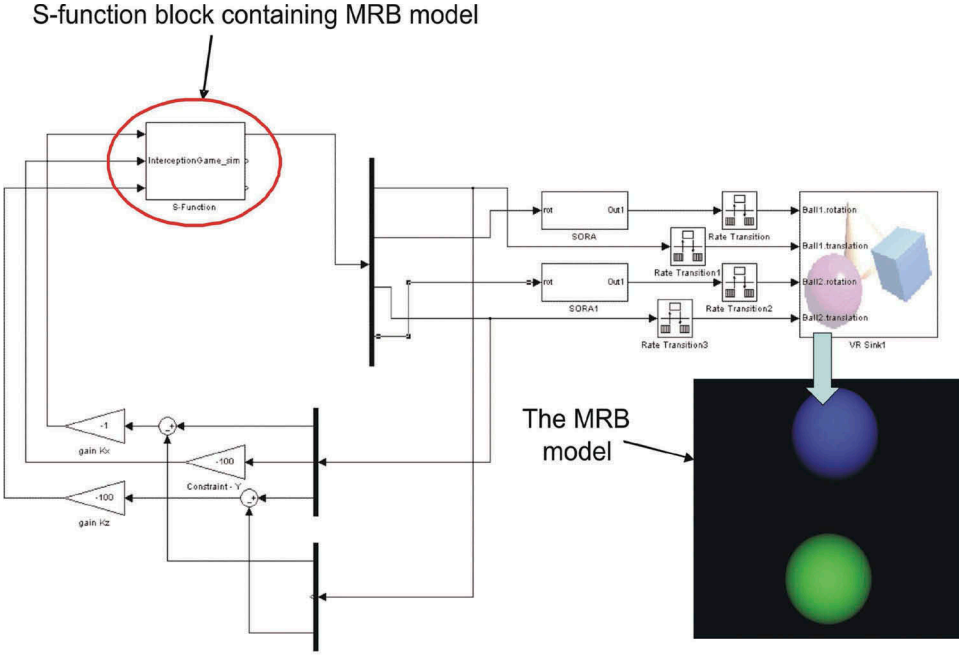

Figure 10. Final output of DyverseRBT for the interception game.

Table 2. MRB hybrid automaton characteristics for multi-contact problems explained in Section 3.8.2.

Model	1 Ball	2 Balls	3 Balls	4 Balls
No. of dynamical locations	1	4	64	4096
No. of contact comp. nodes	0	1	7	63
No. of impact comp. nodes	0	1	7	63
No. of edges	0	11	524	231,284
No. of possible contacts	0	1	3	6

The tool first constructs a list of possible contact situations that occur in the model. Then from this, it derives the number of the different types of locations (contact, impact and dynamical) and assigns the possible edges to each of the locations. Finally, the tool derives the dynamic equations governing the motion for each location from the contact situation assigned to that location – for example, whether the contact is open or closed, stick, trans or slip etc. Finally, the guards and resets are determined. The end result is a complete and precise description of the rigid-body system, as a Matlab structure, in the formulation of an MRB hybrid automaton.

Table 2 shows the number of locations, edges and possible contacts determined by DyverseRBT for each model. Clearly, the number of locations and number of edges between locations rapidly increase with the number of possible contacts. This combinatorial explosion limits the practical size of any model which can be completely described in this way, even with relatively simple rigid-body entities.

4. Application of the MRB hybrid automaton to falsification of safety properties

In this section, we briefly describe DyverseBMC: a procedure for falsifying a safety property over finite time intervals for a rigid-body system against our hybrid automaton abstraction. DyverseBMC has to be understood as a valuable application of the wider modelling and simulation framework that we propose in this paper, which includes the specification of the computational semantics for an MRB system. This specification is given by the MRB hybrid automaton formalism and was explained in the previous section. The automatic generation of the MRB hybrid automaton and its simulation are implemented through the tool DyverseRBT. The main elements of our framework were given in Figure 1.

4.1. *Outline of the falsification procedure*

The problem to solve is formally specified as

- *For a multi-rigid body mechanical system* H_{MRB} *with continuous states* $x(t) \in \mathcal{X}$ *and physical parameters* $p \in \mathbb{R}^{n_p}$, *the property* $\phi(x(t))$ *must hold for all* $x(t)$ *such that* $x(0) \in I \subseteq \mathbb{R}^n$, $p \in P \subset \mathbb{R}^{n_p}$ *and* $t \in T \subset \mathbb{R}$, with $n, n_p \in \mathcal{N}$ some natural numbers, with $\phi(x(t))$ a safety property, I a set of initial conditions, P a set of possible values for the physical parameters (mass, damping etc.), and T some time interval. It is well worth reminding that the safety problem will be reduced to the satisfiability of Boolean formulas, and the approach to follow will be that of falsification.

Verification of hybrid systems is generally a very difficult problem. There are a number of practical restrictions to our method in its current form and in its current implementation:

- *Rigid-body dynamical equations should have low stiffness*. The method of falsification contains a numerical integration component to evaluate ODEs. The consequence is a buildup of error in the continuous states. In order to keep this error low and to ensure that numerical

integration method is stable and efficient, it is necessary that the rigid-body system has relatively low-stiffness dynamics.

- *Solution uncertainty and conservative safety properties*. The numerical integration error is also significant when finding state-space trajectories which violate the safety property. It is uncertain whether a trajectory which runs very close to the boundary of the region in the state space defined by the safety property is safe or unsafe. The integration error may cause, or prevent, the trajectory from crossing the boundary resulting in an incorrect result. To avoid this problem, it is necessary to choose conservative safety properties such that the numerical error will never be large enough over the bounded time interval to cause the bounded model checker to imply safety when the system is unsafe. This still leaves the opposite possibility that the system model checker declares unsafe when the system is safe. However, this can be checked by analysing the counter example and by successive refinement of the safety property. However, the process of successive refinement may never terminate, something well proven even for linear continuous-time systems and linear hybrid systems [57,58,62]. This problem might be solved with the use of alternative numerical time-integration techniques: event-capturing time-stepping schemes (for instance, Ref. [34]), which are not based on the accurate detection of events and are quite robust in contact detection, unlike our event-driven scheme. Note that the work in Ref. [34] proposes a scheme which stabilizes the constraints, which implies that the constraint drift phenomenon – a well-known phenomenon in the systems treated – is avoided.
- *Short-time intervals and small number of location transitions*. The error in the continuous states caused by the numerical integration increases with time. Although we have not characterize this error, we can conclude that the bounded model checker should only be used to check short intervals to keep this error small. We also restrict to small number of transitions from location to location in the MRB hybrid automaton. For example, transitions caused by impacts or changes in friction (stick to slip, for example). The complexity of the problem rises significantly with each transition. For practical solving times, we therefore only consider one or two transitions at the most.

Our falsification approach is based on a methodology akin to BMC described in Section 1. A logical formula is constructed consisting of Boolean variables and constraints on real variables. This formula is satisfied only by a certain set of trajectories across the continuous state space over finite time intervals.

Algorithm 1. DepthFirstSearch (Falsification of a safety property)

Input: s, *root*
if s is a dynamical location **then**
  $(safe, roots) \leftarrow ExploreDynamicalLocation(root, f_s(x), Dom_s, \phi)$
else
  $ExploreCompNode(root, s)$
end if
for all $\varphi \in roots$ (for all formula in *roots*) **do**
  $\varphi_{s'}$ = root formula for location s' = root formula for location
  $safe \leftarrow DepthFirstSearch(s', \varphi_{s'})$
  if $safe = false$ **then**
    return
  end if
end for
Output: variable *safe* (if $safe = false$, the system is unsafe)

A lazy SMT solver [74,79] is used to find a solution to this formula which falsifies the safety property. If no falsifying solution exists, then we can conclude that there are no trajectories amongst that set which are unsafe over the time interval. We then move to a different set of trajectories and repeat the search for unsafe trajectories.

The overall procedure is shown in Algorithm 1. The algorithm uses a depth-first-search method to build possible discrete evolutions, which are then checked for safety. More specifically, the discrete evolution is checked for the existence of a continuous trajectory that starts in the initial conditions, visits each discrete location in order and, at some point, violates the safety property.

For instance, the discrete evolution $s_1 \rightarrow s_2 \rightarrow s_3$ is unsafe, if there exists a continuous trajectory which starts from a set of initial conditions in the domain of the discrete location s_1, then enters the domain of s_2, followed by the domain of s_3, and at some stage enters a region of the state space specified as unsafe.

If the discrete evolution is determined to be safe, then a new location is added to the evolution following the depth-first-search method. This new and longer discrete evolution is then checked for safety, and so on, until an unsafe solution is found, or until some chosen maximum depth is reached.

The safety verification task is conducted using the function *ExploreDynamicalLocation*. The functions *CreateRoot* and *ExploreCompNode* are used to aid in constructing formulas for trajectories that traverse between discrete locations and domains. The purpose and details of these functions are given in the 'Supplemental online material' file. In this file, we also explain the main limitations of our method. One of the most significant limitations is the need to use numerical integration in the Lazy-SMT solver to compute ODE functions.

As we pointed out in Section 1, even though DyverseBMC is not optimized and has limitations, still it represents a valuable application of the proposed alternative modelling framework for the type of systems the MRB hybrid automaton is modelling.

4.2. *Convex and concave constraints*

It is important to give a note on the convex and concave constraints of our falsification procedure. The constraints which appear in the clauses of the formula in our method (see the 'Supplemental online material' file) are not restricted to be convex. In the case of our case study in the next section, both convex and concave constraint surfaces appear regularly in the formulas the Lazy SMT solver is required to solve. This represents a problem. If the Lazy SMT solver returns that the formula is infeasible, how are we to know that this truly is the case? It is possible that the minimization may have become trapped in a local but infeasible minimum, where a feasible minimum also exists. In which case, our determination of infeasibility is dependent on the initial conditions and the result is unsound. Global optimization methods are often statistically based and thus can only assert probable infeasibility.

To solve this problem, we are fortunate in the definition of our MRB hybrid automaton. In Section 2, we imposed the restriction that the contact surfaces are convex. Consequently, as the constraint surfaces in our formula are derived from these contact surfaces, we limit the types of constraint surfaces to either a convex surface or a concave surface – not both in different regions.

For convex surfaces, infeasibility can be known, and the result is sound. For concave surfaces, it follows that the minimal points are on the edge of the allowable range of the variables in our minimization problem. Thus, we in effect know how many local minima exist, and approximately where they are, and can cycle through each one by choosing appropriate initial conditions until one becomes feasible. If none are feasible, then we can return the result infeasible – with the minimum number of infeasible constraints – which we now know to be sound.

4.3. *Implementation*

DyverseBMC is implemented in Matlab (Mathsworks Inc., USA). Boolean satisfiability problems are solved using Matlab's on-board SAT solver. Numerical integration of ODE problems is performed by Matlab's ode45 function. Optimization problems are solved using the NAG toolbox *nag_opt_nlp2_solve* function for minimizing arbitrary smooth objective functions subject to smooth linear or non-linear constraints [87].

The programs that form DyverseBMC and the input files for the examples used in this paper are available at http://staff.cs.manchester.ac.uk/navarroe/research/dyverse/DyverseRBT_BMC.zip.

4.4. *The interception game revisited*

Recall our motivating example introduced in Section 1.3. The aim is to prove that, for all K over an interval $K \in [1, 3]$, where K is the feedback controller gain of ball 2, the game will not be lost within the first two seconds of play. Thus far, we have:

- An MRB hybrid automaton of the system – see Section 3.8.1.
- A set of initial conditions, I, as defined in Section 1.3.
- A property which must hold,

$$y_1(t) > 0, \qquad \forall (K \in [1,3]), \quad \forall (\{\mathbf{x}_1(0), \mathbf{x}_2(0), \underline{\mathbf{x}}_1(0), \underline{\mathbf{x}}_2(0)\} \in I), \quad \forall (t \in [0,2]),$$

where $\mathbf{x}_1, \mathbf{x}_2$ are the state vectors for ball 1 and ball 2, respectively, as defined in Section 1.3, t is time in seconds and the dot denotes derivative with respect to time.

By checking this property, we find whether the game can be sustained for the first 2 s and, by counter example, the solution of K if the game is lost.

The MRB hybrid automaton is constructed for the interception game using the DyverseRBT tool and checked to falsify the safety property using the DyverseBMC tool. The external forces input in the Simulink model (see Figure 10) are added to the automaton manually in the form of symbolic expressions.

We look at how the size of the problem changes as the model checker checks for safety further into the future of the model, and how this change in problem size can affect the performance. Note that the current implementation has not been optimized for performance, and we therefore offer no benchmarks with other SMT-type packages. Furthermore, other SMT-type packages do

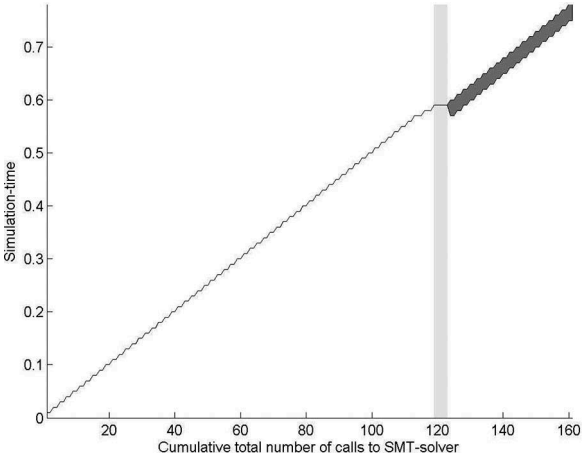

Figure 11. Simulation time for the interception game.

not support numerical integration which is necessary to generate trajectories. We focus instead on how the method of posing the safety property and physical model in the form described in this paper will affect the SMT solver.

The simulation time is the length of the possible trajectories in time, or in other words, the amount of time that has been checked in the model of the interception game. Figure 11 compares the simulation time with the number of calls to the Lazy SMT solver. We point out that the Lazy SMT solver is invoked by the function *Satisfy*, in Algorithms *ExploreDynamicalLocation* and *ExploreCompNode* given in the 'Supplemental online material' file. The *CreateRoots* function adds to the formula a possible range for the simulation time. This divergence of simulation time is represented by a dark-grey region. Each edge transition invokes a call to *CreateRoots*, which diverges this range further. The plateau in the light-grey region is caused by calls to satisfy while exploring the computation node, i.e. *Satisfy* is called from Algorithm *ExploreCompNode*. Computation nodes are considered to act over instantaneous moments in time and, therefore, do not advance the simulation time.

Each iteration of the outer-most while-loop in Algorithm *ExploreDynamicalLocation* increments the simulation time by a time step. During an iteration, the Lazy SMT solver is asked to solve at least $N + 2$ formulas, where N is the number of edges of the dynamical location. Specifically, the Lazy SMT solver is called N times to check if a trajectory exists which enters any of the N guard sets (it is called once for each of the N guards), then once to check if a trajectory is in the domain, and if it is, again to check for unsafe trajectories.

When exploring a computation node, the Lazy SMT solver is called at each iteration of the contact force to test if the new iteration has any non-convergent solutions. After all possible contact-forces have converged, the solver is called a further N times to check for solutions that enter a guard, where as before N is the number of edges leaving the computation node.

The procedures of DyverseBMC are premised on adding new clauses to a formula as new time steps are added and contact situations change. Figure 12 shows the size of the formulas given to the Lazy SMT solver with respect to the simulation time. The size is judged in terms of the number of clauses with one and two literals in the formulas. The number of clauses is broadly stable while exploring dynamical locations. The sudden increase occurs when the computation node is being explored. Thereafter, when exploring the dynamical location, the size of the formulas remains stable. It is primarily the occurrence of contact in the model which increases the difficulty of the problem.

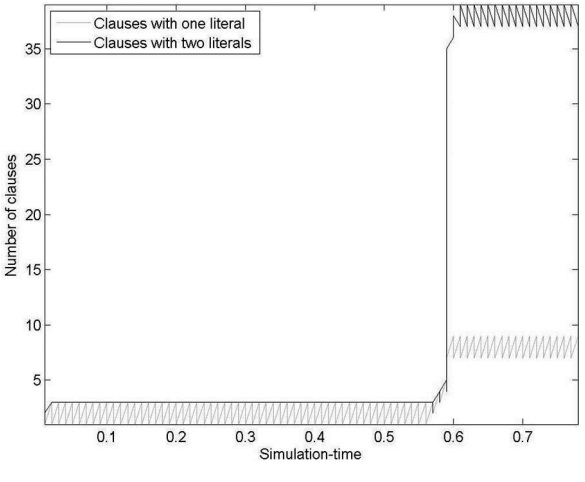

Figure 12. Number of clauses in formula versus simulation time for the interception game.

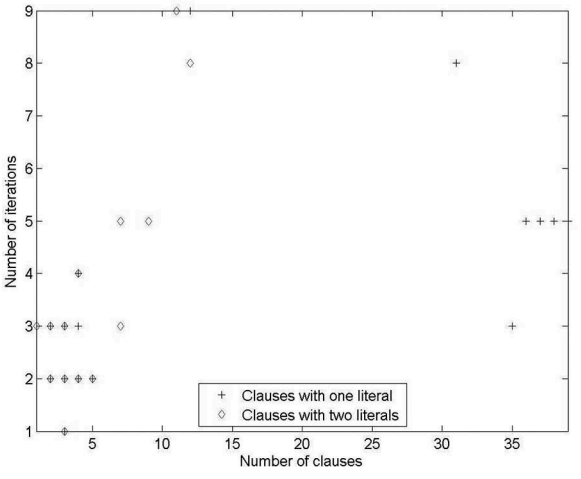

Figure 13. Number of iterations to solve a formula versus number of one and two literal clauses in formula for the interception game.

The difficulty of finding a solution to a given formula grows as the number of clauses increases. The number of single-literal clauses is less significant than clauses with multiple literals. Single-literal clauses only add new constraints for the constraint solver. However, clauses with multiple literals (e.g. $a \vee b$) increase the number of possible Boolean abstractions which might satisfy the formula. Figure 13 shows the number of iterations in Algorithm *Satisfy* (see the 'Supplemental online material' file) required to solve (or prove infeasible) a formula with respect to the number of one-literal and two-literal clauses in that formula. Evidently, the number of iterations increases more dramatically with the number of two-literal clauses compared to single-literal clauses.

In the 'Supplemental online material' file, we also include a multiple contact problem to explore the limitations of DyverseBMC.

5. Conclusions

In this paper, we have presented a novel computational model referred to as the MRB hybrid automaton to fully describe the transitions and operation modes present in a general class of mechanical systems with impacts and friction. Significantly, our computational model can accommodate multiple contacts. This extends the already existing results related to the modelling of systems with impacts by using a class of hybrid automata referred to as MRB hybrid automata. One of the chief characteristics of our model is the inclusion of computation nodes to calculate the contact forces. Each computation node consists of a set of non-dynamical discrete locations, discrete transitions and guards between these locations, and resets on transitions, which can account for the energy transfer not explicitly considered within the rigid-body formalism. Based on the MRB hybrid automaton formalism, we propose a specification of the computational semantics for a class of MRB systems. The automatic generation of the MRB hybrid automaton and its simulation has been implemented through the tool DyverseRBT. The proposed MRB hybrid-automaton-based modelling framework is well suited for the automatic generation of simulation models and the formal verification of dynamical properties of realistic mechanical systems. Due to the complexity involved, building these models or verifying these systems manually is laborious and could be eased by automatic computational techniques as the ones presented in our work.

Acknowledgements

The authors gratefully acknowledge the efforts of the anonymous reviewers and the editor, who gave valuable comments to improve the paper. We are also grateful to Prof. Pieter J. Mosterman for pointing us out the interesting idea of mythical modes. This has been very helpful for building the MRB hybrid automaton.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of the UK: [Grant Number EP/I001689/1] ('DYVERSE: A New Kind of Control for Hybrid Systems'), and the Research Councils UK (RCUK): [Grant Number EP/E50048/1].

References

[1] R. Alur, C. Courcoubetis, T.A. Henzinger, and P.H. Ho, *Hybrid Automata: an Algorithmic Approach to the Specification and Verification of Hybrid Systems, in Hybrid Systems*, R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, eds., LNCS, Vol. 736, Springer, Berlin, 1993.

[2] T. Henzinger, *The theory of hybrid automata*, *Proceedings of the 11th IEEE Symposium of Logic in Computer Science*, Vol. 1, 1996, pp. 278–292.

[3] K.H. Johansson, M. Egerstedt, J. Lygeros, and S. Sastry, *On the regularization of Zeno hybrid automata*, Syst. Control. Lett. 38 (1999), pp. 141–150. doi:10.1016/S0167-6911(99)00059-6

[4] J. Lygeros, K.H. Johansson, S.N. Simic, and S.S. Sastry, *Dynamical properties of hybrid automata*, IEEE Trans Automat Contr 48 (2003), pp. 2–17. doi:10.1109/TAC.2002.806650

[5] V. Acary and B. Brogliato, *Numerical Methods for Nonsmooth Dynamical Systems*, Springer-Verlag Heidelberg, Berlin, 2008.

[6] P. Alart and A. Curnier, *A mixed formulation for fictional contact problems prone to newton like solution methods*, Methods Appl. Mechanics Eng. 92 (1991), pp. 353–375. doi:10.1016/0045-7825(91)90022-X

[7] E. Bayo and R. Laursen, *Augmented lagrangian and mass-orthogonal projection methods for constrained multibody dynamics*, Nonlinear Dyn. 9 (1996), pp. 113–130. doi:10.1007/BF01833296

[8] A. Blumentals, B. Brogliato, and F. Bertails-Descoubes, *The contact problem in Lagrangian systems subject to bilateral and unilateral constraints, with or without sliding Coulomb's friction: a tutorial*, Multibody Syst. Dyn. 38 (2016), pp. 43–76. doi:10.1007/s11044-016-9527-6

[9] B. Brogliato, *Nonsmooth Mechanics: models, Dynamics and Control*, Springer-Verlag, London, 1999.

[10] Y. Hurmuzlu and D. Marghitu, *Rigid body collisioons of planar kinematic chains with multiple contact points*, Int. J. Rob. Res. 13 (1994), pp. 82–92. doi:10.1177/027836499401300106

[11] R. Leine and N. Van De Wouw, *Stability and Convergence of Mechanical Systems with Unilateral Constraints*, Springer-Verlag, Berlin, 2008.

[12] N. Nguyen and B. Brogliato, *Multiple Impacts in Dissipative Granular Chains*, Springer-Verlag, Berlin, 2014.

[13] F. Pfeiffer, *Mechanical Systems Dynamics*, Springer Heidelberg, Berlin, 2008.

[14] C. Studer, *Numerics of Unilateral Contacts and Friction*, Springer-Verlag Heidelberg, Berlin, 2009.

[15] H. Kress-Gazit, *Robot challenges: toward development of verification and synthesis techniques [from the guest editors]*, IEEE Robotics and Automation Magazine 18 (2011), pp. 22–23. 10.1109/MRA.2011.942486

[16] H. Kress-Gazit, T. Wongpiromsarn, and U. Topcu, *Correct, reactive, high-level robot control: mitigating the state explosion problem of temporal logic synthesis*, IEEE Robotics and Automation Magazine 18 (2011), pp. 65–74. 10.1109/MRA.2011.942116

[17] A. Bhatia, M. Maly, E. Kavraki, and M. Vardi, *Motion planning with complex goals*, Robotics Automation Magazine, IEEE 18 (2011), pp. 55–64.

[18] X. Ding, M. Kloetzer, Y. Chen, and C. Belta, *Automatic deployment of robotic teams*, IEEE Robotics Automation Magazine 18 (2011), pp. 75–86. 10.1109/MRA.2011.942117

[19] R. Verma and D. Del Vecchio, *Semi-autonomous multi-vehicle safety*, Robotics and Automation Magazine, IEEE 18 (2011), pp. 44–54.

[20] E. Navarro-López, U. Celikok, and N. Sengör, *Hybrid systems neuroscience, Closed-Loop Neuroscience*, A. El Hady, ed., Vol. 1, Academic Press, 2016, pp. 113–129.

[21] R. Goebel, R. Sanfelice, and A. Teel, *Hybrid dynamical systems. robust stability and control for systems that combine continuous-time and discrete-time dynamics*, IEEE Control Systems Magazine (2009), pp. 28–93.

[22] J. Simo and T. Laursen, *An augmented lagrangian treatment of contact problems involving friction*, Comput. Struct. 42 (1992), pp. 91–116. doi:10.1016/0045-7949(92)90540-G
[23] M.D. O'Toole and E.M. Navarro-López, *A hybrid automaton for a class of multi-Contact rigid-body systems with friction and impacts*, in *Proceedings of the 4th IFAC Conference on Analysis and Design of Hybrid Systems, ADHS 2012*, Eindhoven, The Netherlands, June, 2012, pp. 299–306.
[24] E. Navarro-López and R. Carter, *Hybrid automata: an insight into the discrete abstraction of discontinuous systems*, Int. J. Syst. Sci. 42 (2011), pp. 1883–1898. doi:10.1080/00207721.2010.495189
[25] T. Nishida and S. Doshita, *Reasoning about discontinuous change*, *Proceedings of the 6th National Conference on Artificial Intelligence*, AAAI 1987, Vol. 2, Seattle, Washington, AAAI Press, 1987, pp. 643–648.
[26] P. Mosterman and G. Biswas, *A theory of discontinuities in physical system models*, J. Franklin Inst. 335 (1998), pp. 401–439. doi:10.1016/S0016-0032(96)00126-3
[27] P. Mosterman and G. Biswas, *A comprehensive methodology for building hybrid models of physical systems*, Artif. Intell. 121 (2000), pp. 171–209. doi:10.1016/S0004-3702(00)00032-1
[28] E. Navarro-López, *DYVERSE: from formal verification to biologically-inspired real-time self-organising systems.*, *Computation for Humanity – Information Technology to Advance Society*, J. Zander and P.J. Mosterman, eds., chap. 12, CRC Press/Taylor & Francis, 2013, pp. 303–348.
[29] R. Brockett, *On the computer control of movement*, *Proceedings of the IEEE Conference on Robotics and Automation*, Vol. 1, New York, USA, April, 1988, pp. 534–540.
[30] K. Forbus, *Qualitative process theory*, Artif. Intell. 24 (1984), pp. 85–168. doi:10.1016/0004-3702(84)90038-9
[31] B. Kuipers, *Qualitative simulation*, Artif. Intell. 29 (1986), pp. 298–338. doi:10.1016/0004-3702(86)90073-1
[32] M. Egerstedt and R. Brockett, *Feedback can reduce the specification complexity of motor programs*, IEEE Trans Automat Contr 48 (2003), pp. 213–223. doi:10.1109/TAC.2002.808466
[33] E. Woods, *The hybrid phenomena theory*, in *Proceedings of the 12th International Joint conference of Artificial Inteligence*, Vol. 1, 1991, pp. 71–76.
[34] V. Acary, *Projected event-capturing time-stepping schemes for nonsmooth mechanical systems with unilateral contact and Coulomb's friction*, Comput. Methods Appl. Mechanical Eng. 256 (2013), pp. 224–250. doi:10.1016/j.cma.2012.12.012
[35] D. Stewart and J. Trinkle, *An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and Coulomb friction*, Int. J. Numer. Methods Eng. 39 (1996), pp. 2673–2691. doi:10.1002/(ISSN)1097-0207
[36] A. Filippov, *Differential Equations with Discontinuous Right-Hand Sides*, Kluwer Academic Publishers, Dordrecht, 1988.
[37] V. Utkin, *Sliding Modes in Control Optimization*, Springer-Verlag, Berlin, 1992.
[38] H. Elmqvist, F. Cellier, and M. Otter, *Object-oriented modeling of hybrid systems*, in *Proceedings of the European Simulation Symposium*, Vol. 1, Delft, The Netherlands, 1993, pp. 31–41.
[39] S. Mattsson, *On object-oriente dmodelling of relays and sliding mode behaviour*, in *Proceedings of the 13th Triennial IFAC World Congress*, Vol. 1, San Francisco, USA, 1996, pp. 259–264.
[40] S. Bliudze and S. Furic, *An operational semantics for hybrid systems involving behavioral abstraction*, in *Proceedings of the 10th International Modelica Conference*, Vol. 1, Lund, Sweden, March, 2014, pp. 693–706.
[41] J. Liu and E. Lee, *A component-based approach to modeling and simulating mixed-signal and hybrid systems*, ACM Trans. Model Comput. Simul. 12 (2002), pp. 343–368. doi:10.1145/643120
[42] E. Lee and H. Zheng, *Operational semantics of hybrid systems*, in *Hybrid Systems: Computation and Control*, M. Morari and L. Thiele, eds., Vol. 3414, LNCS, Springer-Verlag, 2005, pp. 25–53.
[43] P. Mosterman, F. Zhao, and G. Biswas, *Sliding mode model semantics and simulation for hybrid systems*, in *Hybrid Systems*, Vol. 1567, LNCS, Springer-Verlag, 1999, pp. 218–237.
[44] P. Mosterman and H. Vangheluwe, *Computer automated multi-paradigm modeling: an introduction*, Simulation 80 (2004), pp. 433–450. doi:10.1177/0037549704050532
[45] E. Navarro-López, *Hybrid-automaton models for simulating systems with sliding motion: still a challenge*, in *Proceedings of the 3rd IFAC Conference on Analysis and Design of Hybrid Systems*, ADHS 2009, Vol. 1, Zaragoza, Spain, September, 2009, pp. 322–327.
[46] E. Navarro-López, *Hybrid modelling of a discontinuous dynamical system including switching control*, in *Proceedings of the 2nd IFAC Conference on Analysis and Control of Chaotic Systems*, CHAOS09, Vol. 1, London, UK, June, 2009.
[47] J. Eker, J. Janneck, E. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong, *Taming heterogeneity: the ptolemy approach*, Proc. IEEE 91 (2003), pp. 127–144. doi:10.1109/JPROC.2002.805829
[48] G. Frehse, C. Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, *Spaceex: scalable Verification of Hybrid Systems, in Computer Aided Verification*, G. Gopalakrishnan and S. Qadeer, eds., Vol. 6806, LNCS, Springer-Verlag, Berlin, 2011, pp. 379–395.
[49] A. Girard, A. Julius, and G. Pappas, *Approximate simulation relations for hybrid systems*, Discrete Event Dyn Syst 18 (2008), pp. 163–179. doi:10.1007/s10626-007-0029-9
[50] A. Girard, *Controller synthesis for safety and reachability via approximate bisimulation*, Automatica 48 (2012), pp. 947–953. doi:10.1016/j.automatica.2012.02.037

[51] G. Pola and P. Tabuada, *Symbolic models for nonlinear control systems: alternating approximate bisimulations*, SIAM J Control Optimization 48 (2009), pp. 719–733. doi:10.1137/070698580
[52] P. Tabuada, *An approximate simulation approach to symbolic control*, IEEE Trans Automat Contr 53 (2008), pp. 1406–1418. doi:10.1109/TAC.2008.925824
[53] P. Mosterman, J. Zander, G. Hamon, and B. Denckla, *A computational model of time for stiff hybrid systems applied to control synthesis*, Control Eng Pract 20 (2012), pp. 2–13. doi:10.1016/j.conengprac.2011.04.013
[54] The MathWorks, Inc., *Stateflow and Stateflow Coder User's Guide. For Complex Logic and State Diagram Modeling*, www.mathworks.com/access/helpdesk_r13/help/pdf_doc/stateflow/sf_ug.pdf (2008).
[55] P. Fritzson, *Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica*, Wiley-IEEE Press, Hoboken, NJ, 2011.
[56] C. Brooks, E. Lee, D. Lorenzetti, T. Nouidui, and M. Wetter, *Demo: cyPhySim A Cyber-Physical Systems Simulator*, *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, HSCC, Vol. 1, Seattle, Washington, USA, April, 2015, pp. 301–302.
[57] A. Bhatia and E. Frazzoli, *Resolution-complete safety falsification of continuous time systems*, in *Proceedings of the 45th IEEE Conference on Decision & Control*, CDC, Vol. 1, San Diego, CA, USA, December, 2006, pp. 3297–3302.
[58] A. Bhatia and E. Frazzoli, *Sampling-based resolution-complete algorithms for safety falsification of linear systems*, in *Hybrid Systems: Computation and Control*, M. Egerstedt and B. Mishra, eds., Vol. 4981, HSCC, LNCS, Springer-Verlag, 2008, pp. 606–609.
[59] P. Cheng and V. Kumar, *Sampling-Based Falsification and Verification of Controllers for Continuous Dynamic Systems*, Workshop on Algorithmic Foundations of Robotics VII, Springer International Publishing, Cham, 2006.
[60] S. Sankaranarayanan and G. Fainekos, *Falsification of temporal properties of hybrid systems using the cross-entropy method*, in *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control*, HSCC 2012, Beijing, China, 2012, pp. 125–134.
[61] A. Bhatia and E. Frazzoli, *Incremental search methods for reachability analysis of continuous and hybrid systems*, in *Hybrid Systems: computation and Control*, R. Alur and G. Pappas, eds., Vol. 2993, HSCC, LNCS, Springer-Verlag, Berlin, 2004, pp. 142–156.
[62] A. Bhatia and E. Frazzoli, *Sampling-based resolution-complete safety falsification of linear hybrid systems*, in *Proceedings of the 45th IEEE Conference on Decision & Control*, CDC, Vol. 1, New Orleans, LA, USA, December, 2007, pp. 3405–3411.
[63] E. Plaku, L. Kavraki, and M. Vardi, *Hybrid systems: from verification to falsification by combining motion planning and discrete search*, Formal Methods Syst. Des. 34 (2009), pp. 157–182. doi:10.1007/s10703-008-0058-5
[64] E. Plaku, L. Kavraki, and M. Vardi, *Falsification of LTL safety properties in hybrid systems*, in *Proceedings of the International Conference on Tools and Algorithms for Construction and Analysis of Systems*, TACAS 2009, 2009, pp. 368–382.
[65] A. Zutshi, S. Sankaranarayanan, J. Deshmukh, and J. Kapinski, *A trajectory splicing approach to concretizing counterexamples for hybrid systems*, in *Proceedings of the 52nd IEEE Conference on Decision and Control*, CDC 2013, Florence, Italy, 2013, pp. 3918–3925.
[66] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, *S-TaLiRo: A tool for temporal logic falsification for hybrid systems*, in *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2011*, Lecture Notes in Computer Science, Vol. 6605, Springer, 2011, pp. 254–257.
[67] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu, *Symbolic model checking without BDDs*, in *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 1999*, Lecture Notes in Computer Science, Vol. 1579, Springer, 1999, pp. 193–207.
[68] F. Copty, L. Fix, R. Fraer, E. Giunchiglia, G. Kamhi, A. Tacchella, and M. Vardi, *Benefits of bounded model checking at an industrial setting*, in *Proceedings of the International Conference on Computer Aided Verification, CAV 2012*, Lecture Notes in Computer Science, Vol. 2102, Springer, 2001, pp. 436–453.
[69] M. Fränzle and C. Herde, *HySAT: an efficient proof engine for bounded model checking of hybrid systems*, Formal Methods Syst. Des. 30 (2007), pp. 179–198. doi:10.1007/s10703-006-0031-0
[70] M. Prasad, A. Biere, and A. Gupta, *A survey of recent advances in sat-based formal verification*, Int J Software Tools Technol Transfer 7 (2005), pp. 156–173. doi:10.1007/s10009-004-0183-4
[71] P. Nuzzo, A. Puggelli, S. Seshia, and A. Sangiovanni-Vincentelli, *CalCS: SMT Solving for Non-linear Convex Constraints*, in *Proceedings of the Conference on Formal Methods in Computer-Aided Design*, FMCAD 2010, Lugano, Switzerland, pp. 71–80.
[72] A. Griggio, B. Fondazione, I. Kessler, and V. Sommarive, *A practical approach to satisfiability modulo linear integer arithmetic*, J. Satisfiability Boolean Modeling and Computation 8 (2012), pp. 1–27 Boolean Modeling and Computation.

[73] J. Rushby, *Harnessing disruptive innovation in formal verification*, in *Proceedings of the 4th IEEE International Conference on Software Engineering and Formal Methods*, SEF M *2006*, 2006, pp. 21–30.
[74] R. Sebastiani, *Lazy satisfiability modulo theories, Journal on Satisfiability*, Boolean Modeling Computat. 3 (2007), pp. 141–224.
[75] M. Fränzle, C. Herde, T. Teige, S. Ratschan, and T. Schubert, *Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure, Journal on Satisfiability*, Boolean Modeling Computat. 1 (2007), pp. 209–236.
[76] A. Eggers, N. Kalinnik, S. Kupferschmid, and T. Teige, *Challenges in constraint-based analysis of hybrid systems*, in *Proceedings of the International Workshop on Recent Advances in Constraints*, Lecture Notes in Computer Science, Vol. 5655, Springer, 2009, pp. 51–65.
[77] G. Audemard, M. Bozzano, A. Cimatti, and R. Sebastiani, *Verifying industrial hybrid systems with mathsat*, Electron. Notes Theor. Compu. Sci. 119 (2005), pp. 17–32. proceedings of the 2nd International Workshop on Bounded Model Checking (BMC 2004). doi:10.1016/j.entcs.2004.12.022
[78] A. Bauer, M. Pister, and M. Tautschnig, *Tool-support for the analysis of hybrid systems and models*, Proceedings of the Conference Design, Automation and Test in Europe, DATE. 2007 (2007), pp. 924–929.
[79] A. Bauer, M. Leucker, C. Schallhart, and M. Tautschnig, *Don't care in SMT: building flexible yet efficient abstraction/refinement solvers*, Int J Software Tools Technol Transfer 12 (2010), pp. 23–37. doi:10.1007/s10009-009-0133-2
[80] A. Cimatti, S. Mover, and S. Tonetta, *Efficient scenario verification for hybrid automata*, in *Proceedings of the International Conference on Computer Aided Verification, CAV2011*, Lecture Notes in Computer Science, Vol. 6806, Springer, 2011, pp. 317–332.
[81] L. Bu, A. Cimatti, X. Li, S. Mover, and S. Tonetta, *Model checking of hybrid systems using shallow synchronization*, in *Proceedings of the International Conference on Formal Techniques for Distributed Systems*, Lecture Notes in Computer Science, Vol. 6117, Springer, 2010, pp. 155–169.
[82] S. Kong, S. Gao, W. Chen, and E. Clarke, *dReach: -Reachability Analysis for Hybrid Systems*, in *Proceedings of the International Conference on Tools and Algorithms for Construction and Analysis of Systems*, TACAS 2015, London, UK, 2015.
[83] S. Gao, S. Kong, and E. Clarke, *dREal: an SMT solver for nonlinear theories over the reals*, in *Proceedings of the International Conference on Automated Deduction - CADE-24*, Lecture Notes in Computer Science, Vol. 7898, Springer, 2013, pp. 208–214.
[84] S. Gao, S. Kong, and E. Clarke, *Satisfiability Modulo ODEs*, in *Proceedings of the International Conference on Formal Methods in Computer-Aided Design*, FMCAD 2013, Portland, Oregon, USA, 2013, pp. 105–112.
[85] A. Van Der Schaft and H. Schumacher, *An Introduction to Hybrid Dynamical Systems, Lecture Notes in Control and Information Sciences*, Vol. 251, Springer-Verlag, London, 2000.
[86] J.S. Pang and J. Trinkle, *Complementarity formulations and existence of solutions of dynamic multi-rigid-body contact problems with coulomb friction*, Math. Programming 73 (1996), pp. 199–226. doi:10.1007/BF02592103
[87] The NAG Library, *The Numerical Algorithms Group (NAG)*, Oxford, United Kingdom, www.nag.com (2013).
[88] C. Studer and C. Glocker, *Solving normal cone inclusion problems in contact mechanics by iterative methods*, J. Syst. Des. Dyn. 1 (2007), pp. 458–467. doi:10.1299/jsdd.1.458

Appendix A. Computation of new contact forces

We briefly review how to compute new contact forces using an iterative Gauss–Seidel relaxation method with proximal point projection, termed the SORPROX method [14,88]. We will not show the derivation of this method, considering only how to implement it using the notation applied in this paper. The interested reader should consult the Ref. [14] for the derivation of this procedure.

Consider a system of multiple rigid bodies with P points of contact between them. Designate j as the index of the contact, and $\Phi_{N,j}, \Phi_{T,j}$ as new estimates of the normal and tangential contact forces (or impulses), respectively, at contact j. The remaining notation in this appendix follows the conventions already established in this paper in Sections 2 and 3. By direct comparison with the SORPROX method, we can derive the following for impact (impulse) and contact computation nodes:

For an impact computation node $\mathcal{I}_k$ (impact situation), the elements $\Phi_{N,j}\Phi_{T,j}$ are

$$\Phi_{N,j} := -\mathbf{A}_{j,j}^{-1}\left(\sum_{n=1}^{j-1}\mathbf{A}_{j,n}\lambda_{N,n}^{(i)} + \sum_{n=j+1}^{N_k}\mathbf{A}_{j,n}\lambda_{N,n}^{(i-1)} + \sum_{n=1,n\neq j}^{N_k}\mathbf{B}_{j,n}\lambda_{T,n}^{(i-1)} - \left(1+\mathbf{e}_{N,j}\right)\dot{g}_{N,j}\right), \tag{A1}$$

$$\Phi_{T,j} := -\mathbf{C}_{j,j}^{-1}\left(\sum_{j-1}^{n=1}\mathbf{C}_{j,n}\lambda_{T,n}^{(i)} + \sum_{N_k}^{n=j+1}\mathbf{C}_{j,n}\lambda_{T,n}^{(i-1)} + \sum_{N_k}^{n=1,n\neq j}\mathbf{B}_{n,j}\lambda_{N,n}^{(i)} - \left(1+\mathbf{e}_{T,j}\right)\dot{g}_{T,j}\right), \tag{A2}$$

where the superscript (i) indicates the iteration, i.e. the ith iteration, and $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are obtained from,

$$\begin{pmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{B}^T & \mathbf{C}\end{pmatrix} = \begin{pmatrix}\mathbf{W}_{N,k}^T\mathbf{M}^{-1}\mathbf{W}_{N,k} & \mathbf{W}_{N,k}^T\mathbf{M}^{-1}\mathbf{W}_{T,k}\\ \mathbf{W}_{T,k}^T\mathbf{M}^{-1}\mathbf{W}_{N,k} & \mathbf{W}_{T,k}^T\mathbf{M}^{-1}\mathbf{W}_{T,k}\end{pmatrix},$$

and,

$$\mathbf{W}_{N,k} = \left(\ldots, W_{N,j}, \ldots\right), \quad \mathbf{W}_{T,k} = \left(\ldots, W_{T,j}, \ldots\right), \qquad \forall j \in I_k.$$

For a contact computation node $\mathcal{C}_k$ (sustained contact situation), $\Phi_{N,j}, \Phi_{T,j}$ are

$$\Phi_{N,j} := -\mathbf{A}_{j,j}^{-1}\left(\sum_{j-1}^{n=1}\mathbf{A}_{j,n}\lambda_{N,n}^{(i-1)} + \sum_{N_k}^{n=j+1}\mathbf{A}_{j,n}\lambda_{N,n}^{(i)} + \sum_{N_k}^{n=1,n\neq j}\mathbf{B}_{j,n}\lambda_{T,n}^{(i-1)} + \ddot{g}_{N,j}\right), \tag{A3}$$

$$\Phi_{T,j} := -\mathbf{C}_{j,j}^{-1}\left(\sum_{n=1}^{j-1}\mathbf{C}_{j,n}\lambda_{T,n}^{(i-1)} + \sum_{n=j+1}^{N_k}\mathbf{C}_{j,n}\lambda_{T,n}^{(i)} + \sum_{n=1,n\neq j}^{N_k}\mathbf{B}_{n,j}\lambda_{N,n}^{(i)} + \ddot{g}_{T,j}\right). \tag{A4}$$

Note that we assume here that j is incremented from 1 to P and that the normal forces are computed before the tangential ones.

The contact forces and impulses must be part of the feasible set implied by the complementarity conditions defined in Section 2. A proximal point projection function is used to enforce this. The function returns the closest point in the feasible set and assigns it to our current iteration of the contact forces. Using the specific contact laws in this paper, we have

$$\lambda_{N,j}^{(i)} := \begin{cases}\Phi_{N,i} & \text{if } \Phi_{N,j} \geq 0,\\ 0 & \text{if } \Phi_{N,j} < 0.\end{cases} \tag{A5}$$

$$\lambda_{T,j}^{(i+1)} := \begin{cases}\Phi_{N,j} & \text{if } \Phi_{N,j} \geq 0,\\ 0 & \text{if } \Phi_{N,j} < 0.\end{cases} \tag{A6}$$

Equations (A1), (A5) and (A6) for an impact computation node, or Equations (A3), (A5) and (A6) for a contact computation node, are iterated until the forces converge:

$$\left|f_{N,j}^{(i)} - f_{N,j}^{(i-1)}\right| < \text{tol} \quad \text{and} \quad \left\|f_{T,j}^{(i)} - f_{T,j}^{(i-1)}\right\| < \text{tol}, \tag{A7}$$

for all P contacts.